Department of Computing
Imperial College London

A high-level framework for efficient computation of performance–energy trade-offs in Markov population models

Anton Stefanek

Submitted in part fulfilment of the requirements for the degree of Doctor of Philosophy in the Department of Computing of Imperial College London

20th January 2014

Declaration of originality

I declare that this thesis was composed by myself, and that the work it presents is my own, except where otherwise stated.

Copyright declaration

Abstract

Internet scale applications such as search engines and social networks run their services on large-scale data centres consisting of tens of thousands of servers. These systems have to cope with explosive and highly variable user demand and maintain a high level of performance. At the same time, the energy consumption of a data centre is one of the major contributors to its operational cost.

This embodies the performance-energy trade-off problem. We need to find configurations which minimise the energy consumed in running important applications in complex environments, but which also allow those applications to run reliably and fast.

In this thesis, we develop a general performance–energy analysis framework that can be used to express complex behaviour in communicating systems and provide a rapid analysis of performance and energy goals. It is intended that this framework can be used both at design time to predict long-run performance and energy consumption of an application in a large execution environment; and at run time to make short-term predictions given current conditions of the environment. In both cases the rapid model analysis permits detailed what-if scenarios to be tested without the need for expensive experiments or time-consuming simulations.

The major contributions of this thesis are:

(i) development of the Population Continuous-Time Markov Chain (PCTMC) representation as a low-level abstraction for very large performance models,

(ii) development of rapid ODE analysis techniques to compute performance-based Service Level Agreements (SLA) and reward-based energy metrics in PCTMCs,

(iii) hybrid extension of PCTMCs that allows models to incorporate continuous variables such as temperature and that permits the specification of systems with time-varying workloads

(iv) an extension of the GPEPA process algebra that can support session-based synchronisation between agents and that can be mapped to PCTMCs, thus giving access to the rapid ODE analysis.

We support the framework with a software tool GPA, which implements all the described formalisms and analysis techniques.

Acknowledgements

First and foremost, I would like to express my sincerest gratitude to my supervisor Dr. Jeremy Bradley. Jeremy has continuously provided me with his invaluable advice and support throughout the past four years. I greatly enjoyed our meetings, always leaving with new insights a unique boost to my motivation. I would also like to thank Jeremy for providing me with financial support through the EPSRC project AMPS on which he was the principal investigator.

Secondly, my thanks go to Dr. Richard Hayden for his willingness to explain many of the theoretical concepts I required to understand in order to develop the contributions in this thesis. I will always highly regard the breadth and depth of his knowledge and ability to clearly convey even the most complicated ideas. None of the contributions in this thesis would be possible without collaboration with Richard and Jeremy.

I would like to thank the following people who have further contributed to the work in this thesis: Dr. William Knottenbelt, my second supervisor and an investigator on the AMPS project for his insightful comments and encouragement; Chris Guenther for the productive collaboration on the GPA tool; Dr. Nigel Thomas for his advice and collaboration on applying the framework of this thesis to the HTCondor system at Newcastle University and Matt Forshaw for giving me efficient access to system logs from HTCondor and for collaboration on analysing the data; Dr. Uli Harder for his ever-present advice and help with energy measurement experiments I performed during my PhD; Gareth Jones for reading a draft of this thesis and for writing hundreds of Wikipedia articles on performance analysis and probability that I have encountered in my research; members of the AESOP research group in the Department of Computing for a pleasant working environment.

I would not be able to pursue my PhD and the studies leading to it without the material and spiritual support from my parents and my sister. I cannot thank them enough for their patience, generosity and encouraging phone conversations. I also want to thank the rest of my family and my friends for their support, especially to Bando for providing me with a comfortable living environment during the final years of my PhD.

Finally, I want to thank Mimi for her love, emotional support and for making the last couple of years so enjoyable.

Contents

List of Figures

List of Tables

Notation

We use the following notation in this thesis (with some exceptions):

$A, B, C, \ldots$	A random variable.
$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \ldots$	A multivariate random variable.
$\mathbb{E}[A]$	Expectation of a random variable.
$\boldsymbol{X}(t), \boldsymbol{Y}(t), \boldsymbol{Z}(t), \ldots$	A continuous-time stochastic process with state space that is a subset of $\mathbb{Z}_+^N$.
$(\boldsymbol{X}(t), \mathcal{C}, \boldsymbol{X_0})$	Specification of a PCTMC with state space $\boldsymbol{X}(t)$, a set of transition classes $\mathcal{C}$ and initial populations $\boldsymbol{X}(0)$.
$\boldsymbol{x}(t), \boldsymbol{y}(t), \boldsymbol{z}(t), \ldots$	A vector function with range in $\mathbb{R}^N$.
$h(\boldsymbol{A})$	Random variable specifying a moment, such as $h(\boldsymbol{A}) = A_1 \cdot A_2$ or $h(\boldsymbol{A}) = (A_1 - \mathbb{E}[A_1])^2$.
S	A positive integer constant specifying a scale of a system.
$(\boldsymbol{X}^{(S)}(t), \mathcal{C}^{(S)}, \boldsymbol{X}_{\boldsymbol{0}}^{(S)})$	A PCTMC that is a *rescaled* version of $(\boldsymbol{X}(t), \mathcal{C}, \boldsymbol{X_0})$ as defined in Section 3.6.
$\widetilde{\mathbb{E}}[h(\boldsymbol{X}(t))]$	An approximation to $\mathbb{E}[h(\boldsymbol{X}(t))]$ usually obtained as a solution to a set of ordinary differential equations such as those defined in Section 3.4
$Agent$	An agent state identifier in a process algebra model. When suitable we use abbreviations such as A for convenience.
$X_C(t)$	The coordinate of a PCTMC state $\boldsymbol{X}(t)$ corresponding to the population of an agent C (assuming the PCTMC is derived from a process algebra model).
$\overline{X_i}(t)$	Shorthand for the accumulated population X_i until time t, that is $\int_0^t X_i(u)\mathrm{d}u$.
$\overline{h(\boldsymbol{X})}(t)$	Shorthand for the accumulated product specified by $h(\boldsymbol{X})$ until time t, that is $\int_0^t h(\boldsymbol{X}(u))\mathrm{d}u$.
$\boldsymbol{\mathcal{X}}(t), \boldsymbol{\mathcal{Y}}(t), \boldsymbol{\mathcal{Z}}(t), \ldots$	A continuous-time stochastic process with state space that is a subset of $\mathbb{R}^M$.
$\underline{s}, \underline{S}$	A socket / set of sockets in the GPEPAc process algebra defined in Chapter 7.
$Agent^{\underline{s}_1,\ldots,\underline{s}_n}$	A GPEPAc agent with sockets.
$\alpha, \beta, \cdots$	A channel variable used in the semantics of GPEPAc
Γ	A channel assignment in GPEPAc transactions.
$[\![G : P^{\underline{s}}(\Gamma), \ldots]\!]$	GPEPAc transaction

> We will often interleave presented definitions and techniques with demonstrations on an example. The corresponding text will be highlighted in the same way as this paragraph.

Simulation plots: Unless stated otherwise, whenever we show a plot of an estimate from stochastic simulation, we use a sufficient number of replications so that confidence intervals are smaller than line thickness of the respective plot.

Chapter 1

Introduction

1.1 Motivation and objectives

Internet scale applications such as search engines and social networks run their services on large-scale data centres consisting of tens of thousands of servers. From a simplified point of view, providers of data centres for such applications have two goals when designing and operating their systems:

Guarantee performance Each user query or an internal request is directed to the appropriate service and executed as a computational task. Providers of the data centre often give a performance guarantee in the form of a *Service Level Agreement* (SLA). SLAs typically assign a minimum probability for the request to finish within a given time, such as "each request will be completed within 0.3s at least 99% of the time" [66].

Minimise energy consumption At the same time, energy consumption of servers in a data centre is one of the main factors in its operational cost [27]. Moreover, data centres are becoming a major contributor to electricity consumption in developed nations and so there are great incentives for providers to reduce their electricity bill and impact on the environment.

There is a technically challenging trade-off to be achieved between these two goals – system configurations which provide energy savings usually reduce the overall system performance, while higher performance configurations tend to have greater energy demand. Therefore, to be able to choose between different configurations of a data centre requires accurate predictions of the resulting performance and the total energy consumption, while taking into account important system SLAs. Ideally such predictions would be provided by a *mathematical model* of the data centre. Using such model would allow the provider of a data centre to evaluate the effects of different configurations without implementing each configuration in a testing environment and performing time consuming and costly benchmarking experiments.

In this thesis, we present a performance analysis framework that will allow us to model such large systems and choose configurations that achieve minimum energy consumption while meeting critical SLAs.

The arrival patterns of requests to a data centre are highly variable and unpredictable and this adds to the modelling challenge. A common way to guarantee an SLA is to heavily over-provision the data centre in order to cope with anticipated peak loads [78]. For example, the total allocated

computational capacity, such as the number of servers, might be increased by 50% above the required capacity at the time of the heaviest load. This has a severe impact on the energy consumption of the data centre, which is often the main component of the overall system cost. Additionally, together with the increasing popularity of data centre based computation, over-provisioning has significant impact on the environment, as most electrical energy used by data centres still comes from non-renewable resources [159, 170].

Data centre architectures provide a number of ways to reduce the energy implications of over-provisioning. These involve a combination of hardware improvements of individual servers and sophisticated algorithms for managing servers and allocating computational tasks. We discuss various energy–performance trade-offs inherent in the different data centre architecture aspects below:

Energy-proportional hardware Depending on the variability of system load, servers in over-provisioned systems are often not fully utilised. In fact, servers spend most of their time at around $30\% - 40\%$ utilisation [27]. Energy-proportional hardware provides a range of operational regimes with decreasing power consumption at the expense of decreased computational capability. The simplest case is for two regimes – normal operation and reduced power consumption for idle periods. Most advances in this direction have been made in case of CPUs, where *Dynamic Voltage and Frequency Scaling* allows the CPU to run at a range of reduced frequencies, thus reducing power consumption and generated heat [105]. This saving can be as much as 70% of the peak power [27]. Similar techniques exist for other hardware components such as RAM and disks, but these have reduced potential power saving and also introduce performance and energy penalties for switching between different regimes.

Dynamic provisioning At the level of a data centre, additional energy proportionality can be achieved by dynamically turning servers on and off, according to the load on the system [167]. In idle periods, servers can be powered down to eliminate their energy consumption. However, in case of sudden increase in demand, there is a significant delay and energy cost to bring the server back up. A compromise can be achieved by a range of "sleep" states where only certain sub-systems are powered down, resulting in much reduced power consumption but also in faster return to an operational state.

Consolidation Due to energy proportionality, servers often achieve their highest energy efficiency when running at full power. At the same time, in order to take full advantage of dynamic provisioning, it is necessary to maximise the number of servers that can enter a sleep state. Often, computational tasks running on a group of servers can be *consolidated* and moved onto a smaller group of servers. One example is virtualisation, where tasks run in *virtual machine* (VM) environments with potentially several VMs on a single server. The VMs can be moved between servers at the expense of temporary performance degradation.

Temperature-aware resource allocation Cooling infrastructure can contribute as much as 50% of the total energy consumption of a data centre [160]. The energy-saving techniques above have an impact on the temperature in a data centre and therefore also on the total energy consumed by the cooling infrastructure. Additionally, there has been progress on

temperature-aware allocation algorithms, which spread system load onto different racks in the data centre in order to minimise the power consumed by cooling infrastructure [111].

Modern data centres implement most of these techniques and give data centre providers access to a large number of different system configurations which can trade energy consumption for performance and vice versa. The resulting system becomes extremely complex, even before considering the details of applications running on the data centre. However, quantitative understanding of the system as a whole is important at all stages of data centre design and operation. For example, at the design stage, data centre providers need to choose a suitable size of their system in order to cope with anticipated loads. Having a quantitative model of the data centre would allow them to rapidly evaluate different configurations without running expensive experiments. Similarly, such a model could be used to assess configuration changes and upgrades once the data center is in operation. In this case the model can be reinforced with parameters obtained from monitoring the real data centre.

In each case, it is essential for the model to be able to quantify the extent to which both of the main goals – performance and energy – have been achieved. Taking all the above considerations into account, we can formulate a number of requirements for this quantitative model:

(i) Deal with **large and complex systems**. We require analysis of systems with tens of thousands of different components. Often, the number of servers is a variable in the configuration and therefore the complexity of the analysis should not be dependent on this number.

(ii) Allow **high-level model descriptions**. The modelled systems are often complex and only possible to understand after being broken down into a number of sub-systems. We require a behavioural language for compositional description of such systems.

(iii) Accurately capture **SLA metrics**. We require the ability to verify SLAs based on passage time probabilities. These SLAs should be described in terms of the behavioural model of the system.

(iv) Capture **energy consumption metrics**. We require efficient methods to compute the total energy consumed by the modelled system.

(v) Jointly consider **temperature and workload**. The computed energy consumption metrics need to be able to take into account the energy consumption of cooling infrastructure, which depends on the environmental temperature which in turn potentially depends on the load on the system and the current configuration.

(vi) Allow **time-dependent** stochastic workloads. Realistic applications need to be able to take into account workloads which are stochastic in nature and vary over time.

1.1.1 Markov population models

Traditionally in performance analysis, a suitable tool for modelling such systems would be a *Markov chain*. Although it is a mathematically very simple stochastic process, it has been successfully applied to capture performance and behaviour in a range of systems. However,

classical Markov chain analysis techniques rely on linear-algebraic operations with complexity at least linear in the number of states of the system. This poses a problem as the number of combinations of states of each of the large number of servers soon explodes beyond the limits of these techniques. Much more suitable is the so-called *Markov population model* approach which treats the system as a Markov chain, but acknowledges the presence of a large number of similarly behaved agents and models their aggregate *populations* instead of considering each agent individually. This representation allows rapid analysis with a derived system of ordinary differential equations (ODEs) the size of which is independent of the scale of the system. For example, in the mean-field analysis [e.g. 88, 89, 42, 31, 33] each population is approximated by a continuous variable from the solution to a system of ODEs. Additional heuristics [e.g. 189, 186, 71, 81, 17] can improve this approximation and also provide ODEs for higher-order moments of populations, such as the variance. Usually, the ODE approximations are related to the original stochastic process by various *convergence results* [e.g. 127] which show that the approximation becomes more accurate as the number of agents increases.

Markov population models and the related analyses originate in physics, chemistry and biology, where the extremely large populations, e.g. of particles, molecules, cells, have always been present. With the increased importance of distributed computation, wireless sensor networks and further massively parallel systems, Markov population models have seen application in computer science and the field of performance analysis [25, 33, 79]. Often, in contrast to applications in natural sciences, the behaviour of individual components, such as servers or wireless sensors, is known and well defined. Traditionally, this resulted in an approach where systems are described in a *behavioural language*, such as a stochastic process algebra, e.g. *stochastic π-calculus* (sπ)[155], *PEPA* [107], *Stochastic Concurrent Constraint Programming* (SCCP) [36] or *Stochastic Petri Nets* [26]. Usually, each of these languages has a defined *semantics* – a translation into an underlying Markov chain, that can be analysed with traditional explicit state space techniques. In some cases, the resulting Markov chain from a behavioural description is equivalent to a Markov population model, and therefore amenable to efficient ODE approximations. Sumpter et al. [171] were one of the first to use this approach and derived a set of *mean-field difference equations* for a discrete time Markov chain described in the *WCCS* process algebra. Further work includes the derivation of ODEs approximating models in PEPA [108, 183], subset of sπ [48], SCCP [41]. These approaches give the set of ODEs as an *alternative semantics* to the Markov chain and in some cases they go on to relate the ODE and Markov chain semantics by showing that the ODEs would be equivalent to applying mean-field techniques to the Markov chain semantics. Hayden and Bradley [99] introduce *GPEPA*, a syntactical extension of PEPA aimed at describing population models. They consider only a single Markov chain semantics and show how to derive ODEs for means and higher-order moments of populations in the Markov chain described by a GPEPA model.

Several extensions to these rapid analysis techniques allow computation of further metrics, such as passage times or rewards. However, to the best of our knowledge, none of the existing results support analysis of SLAs, while being able to accurately capture the evolution of energy consumption over time and simultaneously take into account the temperature of the environment. This is the goal we have set ourselves in this thesis.

1.2 Contributions and thesis outline

In this thesis, we develop Markov population models into a modelling framework for capturing performance–energy trade-offs. We start with the technique of Hayden and Bradley [99], originally derived as an efficient ODE analysis for models described in the GPEPA process algebra. The same authors also show how to use this approach to rapidly evaluate passage time probabilities for SLAs [4]. We generalise this approach to Markov population models and define a common representation called *Population Continuous-Time Markov Chains* (PCTMC). We incorporate various existing heuristics based on moment closures and present a new moment closure specifically for models of computer systems. This greatly improves the accuracy of the ODE analysis of PCTMCs. In doing this, we have a framework which satisfies requirements (i)–(iii). We show how to derive ODEs for moments of accumulated rewards, enabling us to compute detailed energy consumption specifications, as given in requirement (iv). A further extension allows us to include continuous variables in the system, to capture the mutual influence of temperature and the behavioural part of the model, requirement (v), and also allows time-dependent rates that can represent high variability in user demand, requirement (vi). We design a specification language capable of concise description of a number of common interaction patterns. We implement our techniques in an efficient software tool GPA, making the framework readily available to modellers.

Chapter 3: Population Continuous-Time Markov Chains

We summarise the main existing results concerning the analysis of Markov population models. We define a *Population Continuous-Time Markov Chain*, a Markov chain where the state space consists of integer valued populations. We show how existing mean-field and moment closure approximations also referred to as instances of **fluid or ODE analysis** apply to PCTMCs. We show how the method of Hayden et al. [4] can be used to compute passage time probabilities for SLAs in PCTMCs. The chapter ends with an overview of first- and second-order convergence results which justify the use of the mean-field and moment-closure approximations. We argue that PCTMCs can be used as an **intermediate representation** for high-level behavioural description languages, such as GPEPA. In the subsequent chapters, we work on the PCTMC level and therefore make our results applicable to any formalism that can be translated to PCTMCs.

Chapter 4: Improving accuracy of ODE analysis of PCTMCs

Although mean-field analysis becomes more accurate as the system size increases, in practice we require accurate results at arbitrary scales of the system. We investigate the accuracy of mean-field and moment closure approximations. We show a heuristic, using so-called **switch-point distance** that allows us to identify time intervals yielding low accuracy in PCTMC models derived from GPEPA. As a first improvement, we combine the ODE analysis with stochastic simulation in places where there is a low predicted accuracy. We introduce **min-normal closure** – a moment closure for rates containing the $\min$ function based on the normal distribution. Finally, we compare the accuracy of different closures on a large number of model parameters.

Chapter 5: PCTMCs with accumulated rewards

We extend the ODE analysis of PCTMCs with differential equations capturing means and higher moments of **accumulated rewards** in the system. Both **impulse** and **rate** rewards are supported and the resulting framework can be used to model energy consumption as well as additional cost functions such as the cost of switching the state of a server. Importantly, the we can **simultaneously capture SLA metrics** as well as rewards. This lets us formulate a class of optimisation problems addressing the **energy–performance trade-off**. We illustrate the techniques on a larger example of a client–server model with server hibernations.

Chapter 6: Hybrid PCTMCs

We further extend the framework by allowing accumulated rewards to be part of the state space of PCTMC models. This enables us to model for example the **temperature** in a data centre, which reacts to system load, and at the same time include scheduling algorithms which take the temperature into account. We extend ODE analysis techniques to this case. Additionally, we introduce **time-dependent** rates into the framework and thus allow models with time-varying workload.

Chapter 7: High-level specification of transactions

PCTMC is a suitable mathematical formalism to describe very large systems. However, from user perspective, models can often be complicated and difficult to describe manually. We introduce **GPEPAc**, an extension of the GPEPA process algebra, aimed at describing systems with **multi-phase session-based communication**. We give a formal definition of the language, and define a translation to PCTMC. We demonstrate GPEPAc on an example of a **heterogeneous computing cluster**, where we optimise a number of scheduling policies while taking the energy–performance trade-off into account.

Chapter 8: GPA – a tool for rapid analysis of PCTMCs

In order to make the PCTMC framework and extensions from this thesis more accessible, we implemented all the developed techniques in a **software tool GPA**. We describe the architecture of GPA and give an overview of its main features. The tool evolved to support a range of different specification languages, using hybrid PCTMCs as an intermediate representation. GPA also provides a number of efficient solution techniques and its architecture allows fast prototyping of different variations of the ODE analysis.

1.3 Statement of originality and related publications

I declare that this thesis was composed by myself, and that the work it presents is my own, except where otherwise stated.

During the course of my PhD, I co-authored the following publications. Apart from two exceptions [4, 5], I am the first author or joint first author in all these publications.

Journal papers & book chapters

[4] R. A. Hayden, A. Stefanek and J. T. Bradley. **Fluid computation of passage-time distributions in large Markov models**. In: *Theoretical Computer Science* 413.1 (Jan. 2012), pp. 106–141. ISSN: 03043975. DOI: 10.1016/j.tcs.2011.07.017

This paper shows how to use the ODE analysis of GPEPA models to derive distribution functions of a number of different types of passage times. My contribution to this paper is a case study of a customer–service system demonstrating the techniques and implementation in the GPA tool.

[10] A. Stefanek, R. A. Hayden and J. T. Bradley. **Fluid computation of the performance-energy trade-off in large scale Markov models**. In: *SIGMETRICS Perform. Eval. Rev.* 39.3 (2011). DOI: 10.1145/2160803.2160872

This is a short paper where we outline the PCTMC framework and suggest the need for a high-level specification language for session-based communication. This paper later evolved into the work presented in Chapter 7.

[14] A. Stefanek, R. A. Hayden and J. T. Bradley. **Mean-field Analysis of Large Scale Markov Fluid Models with Fluid Dependent and Time-Inhomogeneous Rates**. In: *Annals of Operations Research* to appear (2013)

This is an extended work on hybrid Markov population models [15] that adds time-dependent rates to the framework. This extended version also improves the presentation of the original paper and is the basis for Chapter 6. I wrote the majority of the extensions and the added case study.

[2] J. T. Bradley, M. C. Guenther, R. A. Hayden and A. Stefanek. **GPA - A multiformalism, multisolution approach to efficient analysis of large scale population models**. In: *Theory and Application of Multi-Formalism Modeling*. Ed. by M. Gribaudo and M. Iaconno. IGI Global, 2013. ISBN: 1466646594. DOI: 10.4018/978-1-4666-4659-9

In this book chapter we give an overview of the GPA tool, focusing on the so-called "multi-formalism" features which allow a number of different formalisms to be translated into the intermediate PCTMC representation. I contributed most of the sections concerning the GPA tool. This work is presented in Chapter 8.

International conference & workshop papers

[8] A. Stefanek, R. A. Hayden and J. T. Bradley. **A new tool for the performance analysis of massively parallel computer systems**. In: *Eighth Workshop on Quantitative Aspects of Programming Languages QAPL 2010 March 2728 2010 Paphos Cyprus*. Electronic Proceedings in Theoretical Computer Science (2010). DOI: 10.4204/EPTCS.28.11

In this paper, we present the GPA tool for the first time and use it for the so-called *switch-point analysis* of GPEPA models. I implemented the tool and performed all the experiments, including the switch-point analysis under scaling. Section 4.2 presents this work in an improved form. Hayden contributed most of the theoretical considerations in Section 1.5 of the paper. These results are reviewed as background in Section 3.6 of this thesis.

[9] A. Stefanek, R. A. Hayden and J. T. Bradley. **Fluid Analysis of Energy Consumption using Rewards in Massively Parallel Markov Models**. In: *ICPE'11 - Second Joint WOSP/SIPEW International Conference on Performance Engineering, Karlsruhe, Germany, March 14-16, 2011*. ACM Press, 2011, p. 121. ISBN: 9781450305198. DOI: 10.1145/1958746.1958767

This paper shows how to compute moments of accumulated rewards in Markov population models. I contributed the majority of the paper and implementation of the techniques in the GPA tool. Chapter 5 presents the main results and additionally considers impulse rewards and a new moment closure approximation.

[11] A. Stefanek, R. A. Hayden and J. T. Bradley. **GPA - A Tool for Fluid Scalability Analysis of Massively Parallel Systems**. In: *2011 Eighth International Conference on Quantitative Evaluation of SysTems*. IEEE, Sept. 2011, pp. 147–148. ISBN: 978-1-4577-0973-9. DOI: 10.1109/QEST.2011.26

In this paper we present the parameter exploration and optimisation aspects of the GPA tool. This is demonstrated in Section 8.4.1.

[15] A. Stefanek, R. A. Hayden, M. M. Gonagle and J. T. Bradley. **Mean-Field Analysis of Markov Models with Reward Feedback**. In: *Analytical and Stochastic Modeling Techniques and Applications - 19th International Conference, ASMTA 2012, Grenoble, France, June 4-6, 2012. Proceedings*. Springer, 2012, pp. 193–211. DOI: 10.1007/978-3-642-30782-9_14

This paper shows how to treat accumulated rewards as part of the state space of Markov population models, allowing rates in the underlying stochastic process to depend on rewards. This allows us to jointly capture temperature and other metrics in the system. The extended version of this paper [14] is presented in Chapter 6. This paper also presents, for the first time, a closure for Markov population models with rates with the minimum function, presented in Section 4.4. I am responsible for the main part of the paper and the related implementation in the GPA tool and the experiments used to demonstrate the framework. Hayden contributed the paragraph on regularity conditions in Section 2.1 of the paper and proofs of convergence theorems extended to this framework in Section 3. These are presented in Appendix C.1 of this thesis.

[3] M. C. Guenther, A. Stefanek and J. T. Bradley. **Moment closures for performance models with highly non-linear rates**. In: *Computer Performance Engineering - 9th European Workshop, EPEW 2012, Munich, Germany, July 30, 2012, and 28th UK Workshop, UKPEW 2012, Edinburgh, UK, July 2, 2012, Revised Selected Papers*. Munich: Springer, 2012, pp. 32–47. DOI: 10.1007/978-3-642-36781-6_3

In this paper we experiment with a number of different moment closures in Markov population models. Section 4.5 shows a comparison of the accuracy of three different closures on a large number of examples. Guenther contributed the implementation of a stochastic simulation with confidence intervals and additionally a log-normal closure and an example model in the MASSPA process algebra which are omitted in this thesis.

[5] M. Kohut, A. Stefanek, R. A. Hayden and J. T. Bradley. **Specification and efficient computation of passage-time distributions in GPA**. In: *Proceeding QEST '12 Proceedings of the 2012 Ninth International Conference on Quantitative Evaluation of SysTems*. London, 2012,

pp. 199–200. DOI: 10.1109/QEST.2012.24

In this paper, I worked with the first author on adapting the architecture of the GPA tool in order to enable an extension implementing the *Unified Stochastic Probes* formalism [103].

National workshop papers

[12] A. Stefanek, R. A. Hayden and J. T. Bradley. **GPA - a tool for rapid analysis of very large scale PEPA models**. In: *UKPEW'10, 26th UK Performance Engineering Workshop. 7-8th July, University of Warwick*. 2010, pp. 91–101

This paper presents the GPA tool.

[13] A. Stefanek, R. A. Hayden and J. T. Bradley. **Hybrid analysis of large scale PEPA models**. In: *9th Workshop on Process Algebra and Stochastically Timed Activities (PASTA)*. 2010, p. 29

This paper combines ODE analysis of GPEPA models with stochastic simulation. We present this approach in Section 4.3.

[6] A. Stefanek, M. C. Guenther and J. T. Bradley. **Normal and inhomogeneous moment closures for stochastic process algebras**. In: *10th Workshop on Process Algebra and Stochastically Timed Activities (PASTA'11)*. Ragusa, 2011

In this paper we explore existing moment closure approximations in the context of our PCTMC framework.

[7] A. Stefanek, U. Harder and J. T. Bradley. **Energy Consumption in the Office**. In: *UKPEW'12, 28th UK Performance Engineering Workshop, Edinburgh, UK, July 2, 2012, Revised Selected Papers*. Ed. by M. Tribastone and S. Gilmore. Vol. 7587. Lecture Notes in Computer Science. Springer, 2012, pp. 224–236. ISBN: 978-3-642-36780-9. DOI: 10.1007/978-3-642-36781-6_16

This paper describes an experiment in which we measured energy consumption of office equipment. Guenther contributed the experimental results of different closures applied to a model in the MASSPA process algebra.

[1] J. T. Bradley, M. Forshaw, A. Stefanek and N. Thomas. **Time-inhomogeneous population models of a cycle-stealing distributed system**. In: *UKPEW'13, The 29th UK Performance Engineering Workshop*. 2013

This paper is a preliminary application of our framework to *HTCondor*, a cycle-stealing distributed system. Examples from this paper are presented in Section 9.2. Forshaw provided a framework for obtaining the experimental data from HTCondor deployed at Newcastle University.

Chapter 2

Background

This chapter provides an overview of the relevant background theory that we use and extend in this thesis. The chapter starts with a brief overview of continuous-time Markov chains. Guided by the considerations in Section 1.1, we focus on models that are able to represent systems consisting of a large number of interacting agents. We provide a high-level overview of Markov population models and associated efficient analysis techniques and postpone a detailed and unified treatment of selected results until the next chapter. We show techniques for calculating derived metrics from general continuous-time Markov chains, such as passage-times and accumulated rewards, that are crucial in various applications. We review hybrid models, where continuous rewards can influence the discrete behaviour of the Markov chain. Finally, we survey several existing tools implementing these techniques. Figure 2.1 gives an overview of this chapter and highlights the context of extensions developed in this thesis.

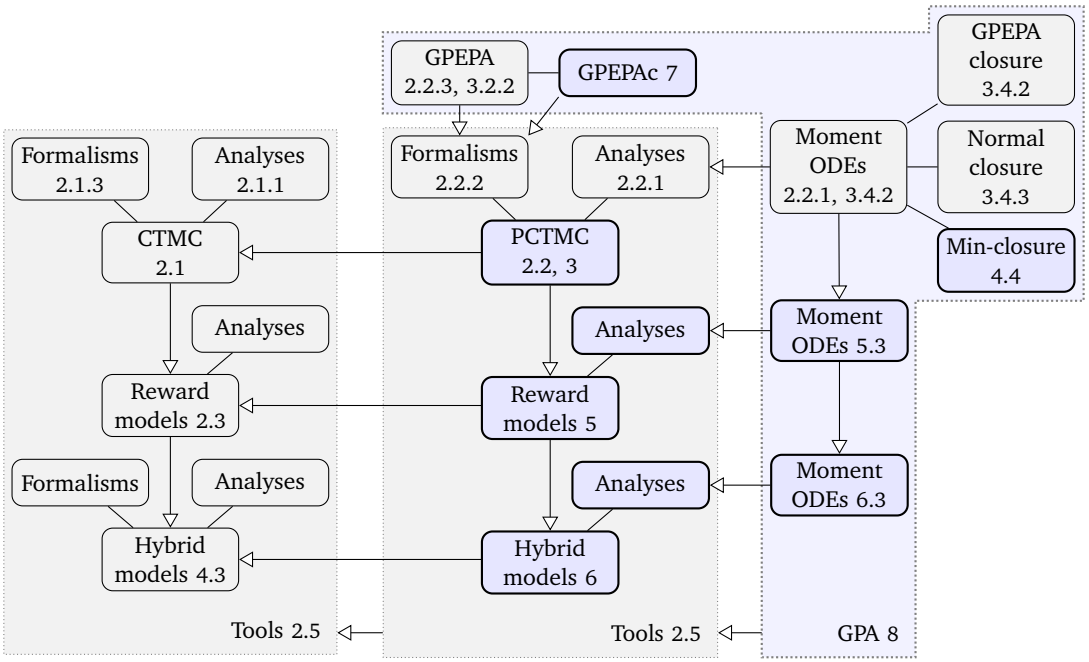


Figure 2.1: Overview of the background and contributions of this thesis. The arrows denote a subclass relationship.

2.1 Continuous-Time Markov Chains

This thesis will be concerned with stochastic processes that are *continuous-time Markov chains* (CTMC), whose treatment can be found in many introductory probability textbooks, [e.g. 163, 150]. We say that a stochastic process $\{X(t)\}$ indexed by a real variable $t \in \mathbb{R}_+$ and taking values in a countable state space $\mathbb{S}$ is a CTMC if it satisfies the *Markov property*, that is if for all $t, \Delta t \in \mathbb{R}_+$,

$$\mathbb{P}\left(X(t+\Delta t) = k \mid X(t) = j, X(u) = x(u), 0 \leq u < t\right) = \mathbb{P}\left(X(t+\Delta t) = k \mid X(t) = j\right).$$

One way to construct a CTMC is to consider a stochastic process with the following behaviour every time it enters a state $s_i \in \mathbb{S}$:

1. the time it stays in the state is exponentially distributed with rate v_i,
2. the next state s_j it enters is distributed according to a discrete distribution with probabilities p_{ij}, $\sum_{j \neq i} p_{ij} = 1$.

Let $p_{ij}(t) = \mathbb{P}(X(t) = j \mid X(0) = i)$. It is possible to show that this construction guarantees that the following two limits exist:

$$\lim_{\Delta t \to 0} \frac{1 - p_{ii}(\Delta t)}{\Delta t} = v_i, \tag{2.1}$$

$$\lim_{\Delta t \to 0} \frac{p_{ij}(\Delta t)}{\Delta t} = p_{ij} v_i =: q_{ij} \qquad i \neq j \tag{2.2}$$

and that

$$p_{ij}(t+s) = \sum_{i \in \mathbb{S}} p_{ik}(t) p_{kj}(s)$$

for all $s, t \in \mathbb{R}_+$.

Further manipulation yields the *Kolmogorov's forward equations*, which characterise the time-evolution of the state probabilities in the CTMC as an ordinary differential equation:

$$\frac{\mathrm{d}}{\mathrm{d}t} p_{ij}(t) = \sum_{k \neq j} q_{kj} p_{ik}(t) - v_j p_{ij}(t)$$

with initial conditions

$$\begin{aligned} p_{ij}(0) &= 0, \qquad i \neq j \\ p_{ii}(0) &= 1. \end{aligned}$$

We can set the *generator matrix* $Q = (q_{ij})$, where $q_{ii} = -v_i$ and define the matrix $P = (p_{ij})$. The above differential equation in matrix form is

$$\frac{\mathrm{d}}{\mathrm{d}t} P(t) = P(t) Q, \tag{2.3}$$

$$P(0) = I$$

where I is the $|\mathbb{S}| \times |\mathbb{S}|$ identity matrix. Usually, we are interested in the probability of $X(t)$ being in a state given a distribution of the state at time $t = 0$, with probabilities in a vector $\boldsymbol{p_0}$. We can reduce the above equation to a single vector $\boldsymbol{p}$:

$$\frac{\mathrm{d}}{\mathrm{d}t}p(t) = \boldsymbol{p}(t)Q \tag{2.4}$$

$$\boldsymbol{p}(0) = \boldsymbol{p_0} \tag{2.5}$$

2.1.1 Analysis techniques

Solution to Equation 2.3 can be expressed as

$$P(t) = \exp(Qt) = \sum_{n=0}^{\infty} \frac{(Qt)^n}{n!}.$$

Using this solution directly yields a numerically unstable algorithm due to both positive and negative entries in the matrix Q. An improvement can be provided by the identity

$$\exp(Qt) = \lim_{n\to\infty} (I + Qt/n)^n.$$

Further improvement is provided by *uniformisation* of $X(t)$ [e.g. 163]. This transforms the CTMC into an equivalent one with additional self-transitions in each state that guarantee that the new holding rates $v_i' = v$ are all equal to the largest v_i in the original chain. The state probabilities can be then expressed as

$$p_{ij}(t) = \sum_{n=0}^{\infty} \left(Q'\right)^n_{ij} \exp(-vt) \frac{(vt)^n}{n!}$$

where $Q' = I + Q/v$. Numerical evaluation of this summation is stable and provides means to control the resulting error of the approximation.

Another possibility is to numerically integrate the set of ODEs in Equation 2.3, for example using the Runge-Kutta algorithm. A common problem with all the presented methods is that they explicitly represent each state of the CTMC. The computational cost therefore depends at least linearly on the size of the state space $|\mathbb{S}|$. This limits their applicability, as even simple CTMC models can have a very large number of states.

2.1.2 Tackling large state spaces

The presented construction of a CTMC can be used in discrete event simulation. Several variants of the *Gillespie algorithm* [82] can be used to generate individual simulation traces of a CTMC. It is often possible to avoid directly evaluating the full state space and the simulation can consider only states encountered throughout each trace. However, in order to obtain accurate estimates of the transient state probabilities, a large number of traces have to be evaluated.

Several symbolic techniques address the problem of having to explicitly store the state-space of the CTMC. The generator matrix of the CTMC can be efficiently stored as a *Multi-terminal Binary Decision Diagram* [74, 106], used in tools such as PRISM [129] and Möbius [65]. A class of CTMCs can be represented as a *Stochastic Automata Network* [153], where the generator matrix can be efficiently derived with tensor algebra. Further implementation optimisations can be used to tackle large state spaces, such as using disks instead of memory and parallelising the computation [121]. However, all of these techniques still have to explicitly represent the probability vector of the CTMC and are therefore not applicable to systems composed of a large number of interacting agents.

2.1.3 Process algebras

In practice, CTMCs are often not described directly but through a high-level language that has a defined translation to a CTMC. Examples include Stochastic Petri-nets [26] and stochastic process algebras such as PEPA [107], stochastic π-calculus [155] and stochastic concurrent constraint programming [36]. Process algebraic descriptions are attractive for modelling real systems. They offer a user-friendly language and are thus accessible to a wider community of modellers. Perhaps the main benefit of describing a system in a process algebra is the *compositionality* property – it is possible to define a system from a number of subcomponents, which can in turn be compositions of smaller components. This allows easy extensions and re-use of existing models. However, the compositionality often results in a combinatorial explosion of the resulting state space as shown in an example below.

2.1.4 PEPA process algebra

In this thesis we will use a variant of the PEPA process algebra. PEPA has a proven history of being applied to a number of different domains, such as mobile networks [73], web servers [44] and robot control [85]. We briefly introduce the syntax of PEPA and its CTMC semantics. We only look at a subset of the language and ignore passive rates and action hiding. A detailed description of PEPA can be found in the original book by Hillston [107]. The main building blocks of PEPA models are agents defined by the following syntax:

$$S := (\alpha, r).S \mid S + S \mid C_S$$
$$P := P \underset{L}{\bowtie} P \mid (P|P) \mid P[n] \mid S \mid C_P$$

The variable S stands for sequential agents and P for parallel agents. The symbol $\underset{L}{\bowtie}$ is the synchronisation operator and L is a set of action labels, $|$ is a shorthand for $\underset{\emptyset}{\bowtie}$ and $P[n]$ for

$$\underbrace{P| \cdots |P}_{n}.$$

The label α is an action label and $r \in \mathbb{R}_+$ is a rate. Named agents can be defined in equations of the form $C \stackrel{def}{=} P$ where C is an agent label. Each PEPA model consists of a number of such definitions and one parallel agent, the *system equation*.

Informally, the semantics of a PEPA model can be described as:

Prefix A sequential agent $(\alpha, r).S$ can perform an action α and after an exponential duration with parameter r evolve into another sequential agent S.

Choice An agent $P + Q$ can perform all the actions of sequential agents P and Q, where the exponentially distributed durations are raced against each other.

Constant Any reference to a named agent C stands for the definition of the agent in one of the equations of the model.

Cooperation An agent $P \underset{L}{\bowtie} Q$ created as a parallel composition of two agents synchronising on a set of actions L allows both P and Q to independently perform any action that is not in L. Actions in L must be executed simultaneously, with both agents changing state accordingly.

Formally, the semantics of a PEPA model is described as a labelled transition system in Figure 2.2. The semantics is inductively defined on the structure of agents and gives all possible transitions $P \xrightarrow{(\alpha, r)} P'$ for each state P of the system equation. Each such transition represents a transition between states P and P' at rate r in the resulting CTMC. The initial state corresponds to the state representing the system equation. The semantic definition uses the concept of *apparent rate* $r_\alpha(P)$, that is the total rate at which an agent P can be seen to perform an action α. A sequential prefix agent can be observed to perform an action α only when α is included in the prefix. If an agent is a choice $P + Q$, then α can be observed either on the P or Q part and so the apparent rate is the sum of the individual apparent rates. PEPA uses the so-called bounded capacity semantics for cooperation, which means that the apparent rate of a cooperation agent $P \underset{L}{\bowtie} Q$ is the minimum of the individual apparent rates if α is in L. The full definition of the apparent rate function $r_\alpha(P)$ is as follows:

$$
\begin{aligned}
r_\alpha((\beta,\, r).P) &:= \begin{cases} r & \text{if } \beta = \alpha \\ 0 & \text{if } \beta \neq \alpha \end{cases} \\
r_\alpha(P + Q) &:= r_\alpha(P) + r_\alpha(Q) \\
r_\alpha(P \underset{L}{\bowtie} Q) &:= \begin{cases} \min(r_\alpha(P),\, r_\alpha(Q)) & \text{if } \alpha \in L \\ r_\alpha(P) + r_\alpha(Q) & \text{if } \alpha \notin L \end{cases} \\
r_\alpha(C) &:= r_\alpha(P) \qquad\qquad \text{if } C \stackrel{def}{=} P
\end{aligned}
\tag{2.6}
$$

The semantic rule for cooperation in Figure 2.2 defines the rate R as a fraction of the total apparent rate $r_\alpha(E \underset{S}{\bowtie} F)$. The motivation for this is that the total rate is split among all the possible combinations of cooperation between a transition E and a transition in F. The term $r_1/r_\alpha(E)$ can be thought of as the probability that the firing of α corresponds to the specific transition $E \xrightarrow{(\alpha, r_1)} E'$.

2.1.5 Client–server model

As an example, consider a simple client–server model. The system consists of two agent types, clients and servers. Clients can request data from servers, receive data from one of the servers and then perform some independent action with the data. Each server, in addition to providing

Prefix: $$\frac{}{(\alpha,\, r).E \xrightarrow{(\alpha,r)} E}$$

Competitive Choice: $$\frac{E \xrightarrow{(\alpha,r)} E'}{E + F \xrightarrow{(\alpha,r)} E'} \qquad \frac{F \xrightarrow{(\alpha,r)} F'}{E + F \xrightarrow{(\alpha,r)} F'}$$

Cooperation: $$\frac{E \xrightarrow{(\alpha,r)} E'}{E \underset{S}{\bowtie} F \xrightarrow{(\alpha,r)} E' \underset{S}{\bowtie} F}\ (\alpha \notin S) \qquad \frac{F \xrightarrow{(\alpha,r)} F'}{E \underset{S}{\bowtie} F \xrightarrow{(\alpha,r)} E \underset{S}{\bowtie} F'}\ (\alpha \notin S)$$

$$\frac{E \xrightarrow{(\alpha,r_1)} E' \qquad F \xrightarrow{(\alpha,r_2)} F'}{E \underset{S}{\bowtie} F \xrightarrow{(\alpha,R)} E' \underset{S}{\bowtie} F'}\ (\alpha \in S)$$

where $R = \frac{r_1}{r_\alpha(E)} \frac{r_2}{r_\alpha(F)} \min(r_\alpha(E),\, r_\alpha(F))$

Constant: $$\frac{E \xrightarrow{(\alpha,r)} E'}{C \xrightarrow{(\alpha,r)} E'}\ (C \stackrel{def}{=} E)$$

Figure 2.2: CTMC semantics of PEPA.

the data, is susceptible to failure in which case it has to be reset. Figure 2.3 shows a diagram of the model. The individual agent definitions in PEPA are:

$$Client \stackrel{def}{=} (request, r_{request}).Client\_wait \qquad Server \stackrel{def}{=} (request, r_{request}).Server\_get + (break, r_{break}).Server\_broken$$

$$Client\_wait \stackrel{def}{=} (data, r_{data}).Client\_think \qquad Server\_get \stackrel{def}{=} (data, r_{data}).Server$$

$$Client\_think \stackrel{def}{=} (think, r_{think}).Client \qquad Server\_broken \stackrel{def}{=} (reset, r_{reset}).Server$$

The system equation of the model consists of $n_C \in \mathbb{Z}_+$ client agents in parallel, cooperating on actions $request$ and $data$ with a parallel composition of $n_S \in \mathbb{Z}_+$ server agents:

$$Client[n_C] \underset{L}{\bowtie} Server[n_S]$$

where $L = \{request, data\}$.

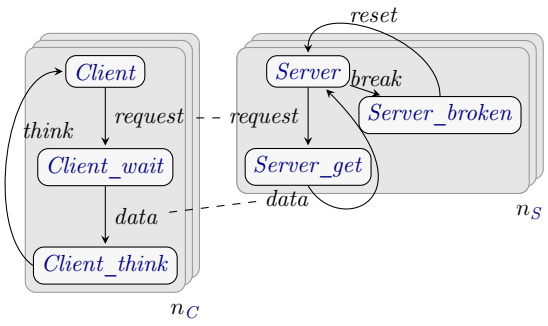


Figure 2.3: Simple client–server model.

For $n_C = n_S = 1$ the CTMC semantics of PEPA generates a CTMC with five states, corresponding to agents:

$$\mathit{Client} \underset{L}{\bowtie} \mathit{Server}, \mathit{Client\_wait} \underset{L}{\bowtie} \mathit{Server\_get}, \mathit{Client\_think} \underset{L}{\bowtie} \mathit{Server}$$
$$\mathit{Client} \underset{L}{\bowtie} \mathit{Server\_broken}, \mathit{Client\_think} \underset{L}{\bowtie} \mathit{Server\_broken}$$

However, the semantics of PEPA causes the state space of this model to grow exponentially with n_C and n_S; in general, the CTMC will have at least $3^{\min(n_C, n_S)}$ states.

2.2 Population models

The large state space in the client–server model is caused by the number of different combinations of states of the n_C individual agents and n_S individual servers. It is possible to reduce the state space if we are not interested in individual agents and instead only want to keep track of the number of clients and servers in their respective states. The states in the original CTMC which have the same count of agents in each of the client and server states can be *aggregated* into superstates. In general, in order to proceed, the resulting aggregated stochastic process has to satisfy a *lumpability* condition [117]. That is, the process has to be a CTMC, in which the rates can be defined as functions of the aggregated state space. In case of PEPA, it is possible to automatically derive such aggregations [84]. For the client–server model, a general state in the aggregated CTMC is of the form

$$(\mathit{Client}[n_1] || \mathit{Client\_wait}[n_2] || \mathit{Client\_think}[n_3]) \underset{\{request, data\}}{\bowtie} (\mathit{Server}[n_4] || \mathit{Server\_get}[n_5] || \mathit{Server\_broken}[n_6])$$

where $n_1 + n_2 + n_3 = n_C$ and $n_4 + n_5 + n_6 = n_S$ (and additionally $n_2 = n_5$).

This CTMC belongs to a subset of CTMCs that is often referred to as *Markov population models*. The analysis of such processes will be the focus of this thesis. Same as the client–server example, each state of a population CTMC (PCTMC) is a finite, integer-valued, vector of N populations, $\boldsymbol{X}(t) \in \mathbb{Z}_+^N$. Transitions in a PCTMC are grouped into a set of *transition classes* $\mathcal{C}$. Each class $c \in \mathcal{C}$ specifies transitions that change a population vector $\boldsymbol{X}(t)$ into $\boldsymbol{X}(t) + \delta_c$ where δ_c is the *change vector* of the class c. The initial values of populations are given by a random variable $\boldsymbol{X_0}$. The rate of c is a function of the populations, $r_c \colon \mathbb{Z}_+^N \to \mathbb{R}$. We postpone a more detailed treatment of PCTMCs until Chapter 3, and here only briefly overview some existing results.

The Kolmogorov forward differential equations, Equation 2.4, for a PCTMC can be written as:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{P}(\boldsymbol{X}(t) = \boldsymbol{x}) = \sum_{c \in \mathcal{C}} [\mathbb{P}(\boldsymbol{X}(t) = \boldsymbol{x} - \boldsymbol{\delta}_c) r_c(\boldsymbol{x} - \boldsymbol{\delta}_c) - \mathbb{P}(\boldsymbol{X}(t) = \boldsymbol{x}) r_c(\boldsymbol{x})] \tag{2.7}$$
$$\mathbb{P}(\boldsymbol{X}(0) = \boldsymbol{x}) = \mathbb{P}(\boldsymbol{X_0} = \boldsymbol{x})$$

2.2.1 Analysis techniques

Standard CTMC techniques can be directly applied to the population case. However, despite the significant state space reduction, population models still suffer from the so-called state space explosion problem. The size of state space of a PCTMC depends on the maximum reachable values of populations in the model. For example, if there are k agent types with s_i states and n_i instances of each, $i = 1, \ldots, k$, the size of the state space is at most

$$\prod_{i=1}^{k} \binom{n_i + s_i - 1}{s_i} = \prod_{i=1}^{k} \frac{(n_i + s_i - 1)(n_i + s_i - 2) \cdots (n_i - 1)}{s_i!}.$$

For example in the client–server model, there are 2 agent types, clients and servers, with 3 possible states each. Setting the initial populations to 100 clients and 50 servers respectively leads to around 10^7 possible states in the PCTMC. Such size quickly reaches the limits of exact methods for solving CTMCs. A large body of research, including this thesis, is interested in the behaviour of such models under a wide range of initial conditions, for example if the number of servers and clients grows by a constant factor S. The traditional methods would restrict the analysis to only small values of S. In this section, we overview analysis techniques which allow analysis of such models for arbitrary values of S. We select a number of results that we further extend in this thesis and present them in a greater detail and in a unified notation in Chapter 3.

Moment closure methods

A very common approach is to use the numerical structure of the state space and transform the forward differential equation for state probabilities, Equation 2.7, into a differential equation describing the evolution of *moments* of populations over time [e.g. 189, 186, 83, 71, 132, 81, 142, 99, 168, 17]. For a moment function $h\colon \mathbb{R}_+^N \to \mathbb{R}$, for example $h(\boldsymbol{X}) = X_i$ specifying a mean of the i-th population, the resulting ODE is (restated in Theorem 1 in the following chapter):

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[h(\boldsymbol{X}(t))] = \sum_{c \in \mathcal{C}} \mathbb{E}\left[\left(h(\boldsymbol{X}(t) + \boldsymbol{\delta}_c) - h(\boldsymbol{X}(t))\right) r_c(\boldsymbol{X}(t))\right] \tag{2.8}$$

The exact form of the right-hand side of this equation depends on the moment function and the rates in the PCTMC. If the rates are linear, it is possible to derive a closed system of ODEs for moments of any order. For example, taking all population means, specified by $h_i(\boldsymbol{X}) = X_i$ $1 \leq i \leq N$, results in a set of N ODEs that can be solved analytically. Taking all means and second-order moments results in a closed system of $N + N(N-1)/2$ ODEs describing the time evolution of means and second-order moments.

However, in most cases the system dynamics requires non-linear rate functions. In such case the terms $\mathbb{E}[h(\boldsymbol{X}(t) + \delta_c) - h(\boldsymbol{X}(t))r_c(\boldsymbol{X}(t))]$ on the right-hand side of Equation 2.8 cannot be expressed as functions of moments or can only be expressed as functions of moments of higher order than the order of $\mathbb{E}[h(\boldsymbol{X}(t))]$ and would thus result in an infinite set of ODEs. One way to solve this problem is to assume a certain distribution of the populations and approximate each arising non-moment expectation or higher-order moment by an expression composed of moments of lower orders, thus resulting in a closed set of ODEs. This process is referred to as a *moment*

closure. Engblom [71] uses Equation 2.8 to derive a system of ODEs for means and higher-order *central* moments. In order to produce a closed system of ODEs, the author sets central moments above a chosen order to be zero. This can be considered as assuming that the distribution of populations shares the moments above a chosen order with a multivariate normal distribution (where the central moments of odd orders are zero). The method is defined on PCTMC models with polynomial rates and other rates have to be dealt with by manually providing a polynomial approximation, such as through a truncated Taylor expansion. We present a related technique using raw moments in Section 3.4.3. A similar approach is taken by Gillespie [81] for models with polynomial rates, later extending to rates that are rational functions of polynomials [146]. In general, we will refer to the class of analysis techniques which approximate moments of populations by a system of differential equations as *ODE analysis* techniques.

Moments can be expressed as functions of lower-order moments under different closures. Singh and Hespanha [166] present a scheme which turns out to be consistent with the populations being jointly lognormally distributed. Krishnarajah et al. [125] use an assumption of a beta-binomial distribution to close the higher-order moments. Ale et al. [17] use a truncated Taylor expansion of the moment equation without further assumptions about the distribution. Hayden and Bradley [99] show a closure specific to a class of PCTMCs with rates that contain instances of the minimum function.

Mean-field approximations

The so-called *mean-field* methods originally emerged from work in the area of statistical physics aimed at capturing the dynamics of complicated particle systems [29, 145, 30]. More recently, the approach has also been applied to the analysis of computer and communication systems [88, 89, 24, 25]. Closely related to our work is the discrete-time mean-field framework introduced by Le Boudec [42, 31] and the continuous-time version given by Bobbio *et al.* [33]. In both of these cases, differential equations provide the limiting mean dynamics in a similar fashion to the moment closures above. The method derives a set of ODEs approximating individual populations. The form of the ODEs is consistent with those from Equation 2.8, under approximation which ignores any co-variance between populations, that is

$$\mathbb{E}[f(\boldsymbol{X})] \approx f(\mathbb{E}[\boldsymbol{X}]).$$

As a formal justification, mean-field methods often consider a number of convergence properties. For a so-called *density dependent* class of PCTMCs, it is possible to construct a sequence of PCTMCs with growing initial populations and moment ODEs with invariant right-hand sides. Using the results of Kurtz [127], it is possible to show that the PCTMC converges to a deterministic limit given by the ODEs as the initial populations grow. We describe these results in greater detail in Section 3.6.

Product-form queueing networks

There are several other efficient techniques for solving specific large PCTMCs without explicitly evaluating the global state space. For example, the structures of Jackson [114] and BCMP [28] queueing networks allow *product form solutions* for steady-state distributions of populations

(queue lengths). However, in many cases these are numerically expensive to compute. For some product-form networks, the *mean value analysis* (MVA) [158] provides a cheaper alternative. The computational cost of MVA depends on the number of jobs in the system. In the presence of multiple job classes, the complexity of the solution grows as the product of populations in different classes. The *method of moments* by Casale [49] addresses this problem and provides a solution technique which depends linearly on the total job population in the network. Thomas and Zhao [178] apply the MVA approach to the analysis of PEPA models. The so-called *Approximate Mean Value Analysis* techniques offer a significant improvement in complexity at the expense of slightly reduced accuracy. Popular examples include the *Linearizer* algorithm by Chandy and Neuse [52] and *Proportional Estimation* algorithm by Schweitzer [164]; both avoid the combinatorial explosion and give an iterative solution that only depends on the number of job classes in the system. The accuracy and numerical properties of these algorithms have been improved in a number of extensions [62, 187]. Unfortunately, to the best of our knowledge, these techniques have not been successfully generalised beyond queueing networks and steady-state metrics.

Diffusion approximations

Another example of an efficient analysis of population models is the *diffusion approximation* [e.g. 126], which are used in many areas such as queueing networks [122, 123] or epidemic modelling [19]. The populations are approximated by a suitably scaled and shifted *Brownian motion* [e.g. 72] – a continuous-time, real-valued stochastic process $E(t)$ such that: (i) $E(0) = 0$ and almost every sample path is continuous, (ii) $E(t)$ has stationary and (iii) independent increments and $E(t) - E(s)$ is normally distributed with mean 0 and variance $\sigma^2(t-s)$ for some $\sigma \in \mathbb{R}$, $0 \leq s < t$. Brownian motion is significantly more tractable than the original stochastic process, for example its stationary distribution has a simple closed-form expression and many transient properties can be obtained. It can be shown that the Markov chain converges to the diffusion approximation as the scale of model increases [128, 100]. The Brownian motion representation can be used to derive a system of ODEs for the covariance matrix of the approximation, which in turn can heuristically justify some of the second-order moment closures mentioned above [100]. We review this approach in Section 3.6.2.

2.2.2 Process algebras for population models

Several process algebras have been extended to be able to conveniently describe population models. Hillston [108] provided an alternative, continuous state space semantics for a subset of PEPA models. This so called *fluid-flow approximation* can be shown to be equivalent to implicitly generating a population model and then applying the mean-field approximation which becomes more accurate as the scale of the system increases [183, 68, 99]. Several further papers extended this type of analysis to a larger subset of the language [43, 68, 99, 101, 183]. Hayden and Bradley [99] defined a variant of the PEPA language, so-called *Grouped PEPA* (GPEPA), which enables a more explicit translation to a population model. The syntax of GPEPA is given in the next section and its detailed PCTMC semantics in Section 3.2.2.

Bio-PEPA by Ciocchetta and Hillston [56] is an extension of PEPA aimed at applying the fluid flow approximation to compositional descriptions of bio-chemical systems. The syntax combines PEPA

with chemical equations. Reactions with arbitrary stoichiometric matrices and rate functions are supported. The semantics of Bio-PEPA generates so-called *CTMC with levels*, which can be thought of as a PCTMC where the values of populations represent discrete levels of molecule concentrations. Bio-PEPA is supported by a software tool [55] and has been applied in biology [16, 54], epidemiology [57], crowd dynamics [140, 141] and collective systems dynamics [138, 139].

Cardelli [48] defines a continuous ODE semantics for *chemical ground form*, a subset of stochastic π-calculus that is equivalent to a class of chemical equations. Models in chemical ground form can be simulated by the *SPiM* tool [152]. Kwiatkowski and Stark [130] introduce *continuous π-calculus*, a process algebra inspired by π-calculus, with a well defined ODE semantics.

In the stochastic concurrent constraint programming (sCCP) formalism [36], the agents communicate through a numerical vector valued *store*. Bortolussi and Policriti [41] associate a set of ODEs to the evolution of the store variables and also provide a translation from ODEs to an equivalent sCCP model. A later extension also derives a set of ODEs capturing variances and covariances of the store variables [35].

2.2.3 Grouped PEPA

In general, there are multiple ways of assigning a PCTMC to a PEPA model. Hayden and Bradley [99] address this ambiguity by a simple syntactic extension, *Grouped PEPA* (*GPEPA*), which introduces a new layer of syntax explicitly specifying the agents and states that will be aggregated in populations.

Formally, GPEPA replaces the system equation of PEPA models by a *Grouped PEPA model*, defined as:

$$G := G \underset{L}{\bowtie} G \;\mid\; G \parallel G \;\mid\; \mathbf{Y}\{P \parallel \cdots \parallel P\}$$

This defines a GPEPA model to be either a PEPA cooperation between two GPEPA models $G \underset{L}{\bowtie} G$ (over a set of actions L), written as $G \parallel G$ if L is empty, or alternatively a labelled grouping of PEPA agents, P, in parallel with each other, where $\mathbf{Y}$ is a *group label*. Agents in each group are separated by the $\parallel$ operator, or alternatively using the $[n]$ notation as for PEPA models. In order to avoid confusion, we will not use the $[n]$ notation for PEPA agents inside a GPEPA group. A Grouped PEPA model is nothing more than a standard PEPA model with, additionally, a structure of labels defining the agents to be aggregated in populations. We assume that labels do not repeat within a single GPEPA model and can be thus used to uniquely identify a group in the model.

For example, the client–server model from Section 2.1.5 can be defined in GPEPA by changing the system equation to

$$\mathbf{Clients}\{\mathit{Client}[n_C]\} \underset{\{request,data\}}{\bowtie} \mathbf{Servers}\{\mathit{Server}[n_S]\}$$

We show a precise translation from GPEPA models to PCTMC in Section 3.2.2.

2.2.4 Transaction-based interactions

All the population-based stochastic process algebras presented above consider only single transition interactions between individual agents. For example, when a user sends a request to a web server, usually there is a sequence of interactions from the web-server to other subsystems (such as a database server) before the user is served the page. Normally, the web server keeps track of the relevant session with the given user. Such a session information is difficult to maintain in the formalisms presented above. For example, in the client–server model, a server cannot keep track of the specific client between the request and data actions. However, in this case it would be straightforward to define a PCTMC of the model with additional virtual populations that would represent a set of agents cooperating in a session, such as a pair of a client and a server.

In Chapter 7, we present a lightweight extension of GPEPA which allows a compositional specification of session-based cooperation. We extend processes with channels which will allow formation of transactions, in a similar fashion to stochastic π-calculus. The channel creation in stochastic π-calculus can be directly used to model transactions. However, none of the existing techniques for efficient analysis of π-calculus can deal with general models – in fact it is exactly the channel creation and forwarding that is omitted from the chemical ground form subset [48].

A similar goal can be achieved by the *Layered Queueing Networks* formalism, which can be analysed by fluid techniques [180]. However, this approach does not allow for the possibility of service forwarding which is essential for example for capturing job allocation in distributed virtualised environments.

2.3 Reward models

So far, the mentioned methods to analyse CTMCs compute properties of the transient distribution of states in a model. Usually, CTMCs are used to evaluate a number of derived metrics. For example, if a CTMC models a single server component (such as *Server* above), measures of interest include the distribution of the total energy consumption up-to time t. This is an instance of an *accumulated reward* in the system and this section gives an overview of the available analysis techniques.

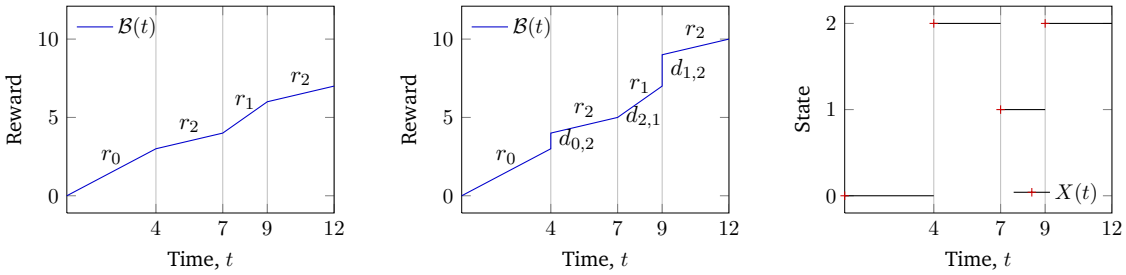
Consider a CTMC $X(t)$ with states from a set $\mathbb{S}$. An *accumulated rate reward* is a continuous-time real-valued process $\mathcal{B}(t)$ defined as

$$\mathcal{B}(t) = \int_0^t r_{X(u)} \mathrm{d}u$$

where r_i, $i \in \mathbb{S}$ are *reward rates* in each state. Additionally, a reward can increase by a constant immediately after a state transition, that is

$$\mathcal{B}(t) = \int_0^t r_{X(u)} \mathrm{d}u + \sum_{i=0}^{K} d_{X(T_i),X(T_i+)}$$

where T_i, $0 \leq i \leq K$, are the times of state transitions, $X(t+)$ is the state immediately after time t and $d_{i,j}$ $i, j \in \mathbb{S}$, are rate constants. Such reward is referred to as an *impulse reward* and can, for



(a) Rate reward

(b) Rate and impulse reward

(c) Underlying stochastic process

Figure 2.4: Rate and impulse accumulated rewards, $\mathcal{B}(t)$, as the underlying process, $X(t)$, evolves.

example, model a cost associated with a transition, such as switching a server state. Figure 2.4 shows an example of the two rewards. Sometimes, further elements of reward accumulation strategies can be considered. For example, the reward can be reset at each state transition (a so-called preemptive restart strategy). In this thesis we only consider rewards of the above form, that is under a so-called preemptive resume strategy, but we allow negative rewards and negative constants r_i and $d_{i,j}$. A good summary of reward strategies can be found in Horváth *et al.* [110].

The combined process $(X(t), \mathcal{B}(t))$ is often referred to as a *Markov reward model* and there is much prior work that analyses the transient distribution of the underlying reward. Most of these techniques are based on numerical methods which require explicit consideration of the entire state space of the associated CTMC. A common approach is to uniformise the CTMC and calculate the total accumulated reward on the underlying discrete-time Markov chain [69, 149, 67, 50]. Telek and Rácz [175] describe an efficient numerically stable algorithm that can compute moments of the accumulated reward for CTMC models with up to 10^6 states (on standard hardware from 2006). Telek et al. [174] consider inhomogeneous Markov reward models. They derive a system of partial differential equations describing the transient distribution of a given reward, which can be reduced to a system of ordinary differential equations for moments of the reward. Inhomogeneous models are also considered by Tijms and Veldman [179]. Horváth et al. [110] provide an implementation of a number of these techniques.

To our best knowledge, all of the existing methods have complexity at least linearly dependent on the number of states and transitions in the CTMC. This prevents them from being applicable to large population models. In Chapter 5, we show an extension of the moment closure method to calculate moments of accumulated rewards in PCTMCs. A similar approach has been suggested in the context of physical chemistry [83]. However, it is limited only to agents of a single type and therefore not applicable to our models where agents from different classes can be in multiple states. Tribastone et al. [181] consider a class of rate rewards in their fluid framework for PEPA. They show how to calculate *action throughput*, the rate at which an action is fired over time, and *capacity utilisation*, the proportion of time an agent is used for a specified action, using ODEs to approximate agent populations. A similar technique is used by Ding [68]. Both of these approaches only consider average values in the steady state of the system. Hayden et al. [103] extend the ODEs for mean populations in GPEPA with ODEs capturing means of so-called

action-counting processes. We show how this technique can be used to compute moments of impulse rewards in PCTMCs.

2.3.1 Passage times and completion times

One of the most useful derived metrics from CTMC behaviour are so-called *passage times* (also called response times). These capture the distribution of the time it takes the model to reach a certain state or a specified sequence of states or transitions. Passage times are often used in practice as part of *Service Level Agreements* (SLAs), such as "a message has to reach its destination within 2 seconds at least 99% of the time". In CTMCs, computation of passage-time distributions is often performed by techniques based on Laplace transforms [93] or via uniformisation [144, 148] where the former are usually less efficient. The uniformisation techniques obtain passage-time distribution through a transformation of the state space. An *absorbing state* is added without modifying the observable behaviour of the model. The passage-time distribution can then be expressed in terms of transient distribution of the added state [144]. Therefore the computation is often at least as complex as computing the transient solution of the CTMC and not applicable to large models [45].

Hayden et al. [4] show how to compute a number of different passage-time classes in GPEPA models. Their method modifies the state space of a model in a similar way to the uniformisation approach above and subsequently applies the ODE analysis. Computed moments of populations can be used to provide approximations to distributions of passage times. We describe the technique in greater detail in Section 3.5, where we also show how it can be extended to a more general class of PCTMCs. Additionally, Hayden et al. [103] present *Unified Stochastic Probes* – a regular-expression-based formalism that allows specification of complex behaviour and state based passage-time measures. These can be automatically translated to moment based measures and therefore the underlying distributions efficiently computed for large models. In a related approach, Bortolussi and Hillston [40] show how to use fluid approximations to analyse single-agent properties described by *Continuous Stochastic Logic* formulae. Ding [68] and Tribastone et al. [181] also consider passage times by applying Little's law to the rewards mentioned above, but they only provide mean measures and only for some types of passage time metrics.

Related to passage times in case of reward models is the so-called *completion time* – the time an accumulated reward reaches a certain target value. Many of the techniques for analysing Markov reward models also consider completion times [e.g. 174]. Horváth et al. [110] derive bounds on the distribution of a completion time using moments of the reward. We show a similar approach in Chapter 5, where the reward is defined alongside a large-scale PCTMC model.

2.3.2 Performance–energy trade-offs

A common application of reward models is to model energy consumption in computer systems, where the goal is to compute system parameters that minimise the total energy consumption. These parameters include the job placement policy in virtualised environments, CPU frequency, rates of switching servers into a sleep state. However, usually the configurations which reduce energy consumption result in a deterioration of the performance provided by the system. This

trade-off, an instance of a multi-objective optimisation problem [136], can be addressed in various ways.

A common approach is to minimise a weighted sum of the mean power or energy consumption and a mean response time, *energy–response weighted sum* (ERWS). Wierman et al. [191] show how to choose an optimal dynamic speed scaling policy that minimises a combination of mean response time and mean energy consumption. Gelenbe and Morfopoulou [80] provide a gradient-descent algorithm to minimise a combination of mean energy consumption and response time in a wired network. Gandhi et al. [76] argue that a more suitable metric to minimise is the *energy-response product* (ERP, or *energy-delay product* [86]), a product of the mean energy consumption and mean response time, which unlike ERWS does not require subjectively chosen weight parameters. They provide optimal policies for management of a multi-server farm, taking into account time-dependent workload and demonstrate their technique on a real trace from the World Cup 1998 website. A similar approach is to maximise "performance per Watt", the inverse of a product of mean power and mean response time [77].

Riska and Smirni [162] quantify power savings in the operation of disk drives under different workloads and give the resulting performance degradation. This allows the modeler to choose the optimal management policy given an acceptable performance degradation. Similarly, Gandhi et al. [78] provide a combination of a predictive and reactive provisioning in a data centre and are able to quantify the energy impact and the number of SLA violations. Ardagna et al. [21] assign a utility function to SLA satisfaction and violation and maximise the total gain (income minus cost) in a virtualised environment. Slegers *et al.* [167] model large server farms with Markov decision processes and evaluate the performance of heuristic allocation strategies. Clark *et al.* [59] model industrial service-oriented systems described in a high level formalism. They translate high level system descriptions into models in the PEPA process algebra and use the efficient fluid analysis techniques to experiment with a large number of system parameter configurations.

In this thesis, we present a framework that can be used to address performance–energy trade-offs in Markov population models, respecting SLAs based on passage time probabilities. Chapter 5 shows how to use the ODE analysis to efficiently calculate moments of rewards in PCTMCs and thus simultaneously calculate both energy consumption and response time CDF in our models. This results in a global optimisation problem with an embedded system of ODEs, Section 5.5, where the objective is to minimise energy consumption and constraints are given as minimum probabilities on passage time CDFs representing given SLAs. Although this problem is too general to be solved analytically, the relatively cheap cost of solving the ODEs numerically leads to efficient approximate solutions. Our approach additionally allows both the model and the SLA response times to be specified in a high-level behavioural language.

2.4 Hybrid models

In this thesis, we consider hybrid models, a generalisation of stochastic reward models. The state space of the underlying stochastic process is extended with continuous variables. These usually evolve deterministically between state changes of the discrete process, according to a system of ordinary differential equations. As opposed to pure reward models, the continuous

variables also affect the discrete behaviour and their values can be used in the corresponding transition rates. There are two common needs for hybrid models. From modelling perspective, continuous variables are often natural elements in the system. On the other hand, the complexity of a population model can be greatly reduced if some of the populations (usually those occurring in great abundance) are approximated as deterministically evolving continuous quantities.

Many of the systems to which CTMC models are applied also consist of continuous variables. For example, in addition to server energy consumption in the client–server model that can be represented as a reward, the servers can generate heat that will affect the overall temperature in the data centre. This temperature is controlled by a set of air conditioning units and also can affect placement of tasks on the servers if a temperature-aware scheduling policy is used. This is an example of a so-called *cyber-physical* system. Many formalisms have been used in the past to model such systems, where the discrete part is captured as a Markov model and the continuous quantities evolve as ODEs over time. The resulting joint stochastic process is an instance of a *Piecewise-Deterministic Markov Process* (PDMP).

Fluid models are an extension of reward models, where the rates are allowed to be negative and the level of the fluid is allowed to influence the behaviour (rates) in the underlying CTMC. The fluid quantity is usually bounded at zero and often additional barriers are introduced, requiring a more sophisticated analysis than reward models. Gribaudo and Telek [90] give an overview of different types of fluid models and the associated solution techniques.

A high-level formalism to describe a class of fluid models is the *Fluid Stochastic Petri Nets* (FSPNs) framework. These are a natural extension of Stochastic Petri nets that introduce places with continuous tokens and arcs with fluid flow [184]. A later extension by Horton et al. [109] allows the level of fluid in continuous places to affect the discrete transitions in the model, while still being amenable to numerical analysis. A further extension by Ciardo et al. [53] is only analysable by discrete-event simulation. The authors describe a simulation algorithm, which is no longer a trivial extension of the simulation of the underlying CTMC and requires approaches similar to that of simulation of non-homogeneous Markov chains. Gribaudo et al. [91] give an automated mapping from a stochastic Petri net with generally distributed transitions into a FSPN. Tuffin et al. [185] provide a comparison between FSPNs and traditional hybrid systems.

There are several process algebras specialised for the modelling of hybrid systems [119]. Galpin et al. [75] introduce HYPE, a process algebra based on PEPA where the evolution of the continuous variables can be defined in a compositional way. Transitions of individual components apply *influences* to continuous variables and the collection of all influences fully determines the evolution of continuous variables at each time. Semantics of a HYPE model is given as a hybrid automaton [104]. HYPE is further extended with stochastic events [37], where the semantics is given as a PDMP.

All the mentioned hybrid modelling techniques require an explicit consideration of the discrete state space of the model, preventing them from being applicable to scalable PCTMC models. In Chapter 6, we show how to adapt the mean-field and moment closure techniques to augment PCTMCs with continuous variables. The continuous variables evolve according to an explicitly given system of ODEs involving populations in the PCTMC and the PCTMC rates can in turn

involve the continuous variables. We show how to derive ODEs capturing moments of such continuous variables and joint moments of continuous variables and populations. Because our approach is ODE based, we cannot enforce boundaries on the continuous variables as is the case in fluid models or FSPNs. We only consider the continuous extension at the level of PCTMCs and will leave a process algebraic compositional description of the continuous behaviour as future work. Additionally, we use a continuous variable to represent time, thus allowing time-dependent rates to be introduced in the model.

We note that hybrid models are often used as a way to deal with large state spaces in population models. The accuracy of mean-field techniques and moment closures is often lower when some of the populations stay small for a longer period of time. For example, in models of genetic pathways in biology, there are often only a few instances of genes that regulate the production of a large number of proteins. Or in the client–server model, a small number of servers can serve a large number of clients. Instead of solving such model as a PCTMC by one of the ODE-based techniques, it is possible to approximate large populations by a continuous variable and then solve the resulting hybrid model exactly, explicitly treating the state space of the small population components. Bortolussi [34] considers such approximations for sCCP models. He formally relates the hybrid system to the PCTMC as a limit when the large populations scale. Hasenauer et al. [94] derive a closed system of differential algebraic equations where equations for the distribution of small populations over time are solved together with equations for moments of large populations.

We will not consider this kind of hybrid approximation in this thesis and will instead try to improve the accuracy of moment closures even for cases with small populations.

2.5 Software tools for population models

There are many tools which support analysis of very large state spaces in performance modelling. Two such popular tools which have good support for explicit state-space analysis are Möbius and PRISM.

The Möbius [65] framework has perhaps the widest user base with implementations of many formalisms, including stochastic process algebras (SPAs), stochastic automata networks (SANs) and generalised stochastic Petri nets (GSPNs). Möbius supports a distributed simulation environment and numerical solvers for models of up to tens of millions of states.

PRISM [129] is a probabilistic model checker which supports low level formalisms such as DTMCs, CTMCs and Markov Decision Processes (MDPs) with an analysis engine based on Binary Decision Diagrams (BDDs) and Multi-Terminal Binary Decision Diagrams (MTBDDs). PRISM can analyse models of up to 10^{11} states, however this can depend heavily on the model being studied and on detailed considerations such as the exact variable ordering in the underlying MTBDD.

Performance tools that support differential-equation based analysis have been primarily designed around stochastic process algebras such as stochastic π-calculus and PEPA. For π-calculus SPiM [151, 152] has long been the standard tool for simulating stochastic π calculus models, but being a simulator it suffers from scalability issues for models with very large populations of components. A recent tool, JSPiM [169], allows for the *chemical ground form* subset of stochastic π-calculus to

be analysed via differential equations. The tools *ipc* [58, 45] and the *Eclipse PEPA plug-in* [182] implement the so-called *fluid translation* [108] to produce sets of differential equations for the stochastic process algebra PEPA.

In the field of biological modelling, tools such as Dizzy [156] and SPiM have been used to capture first-order approximations to system dynamics using a combination of stochastic simulation [82] and differential equation approximation. A recent tool by Gillespie [81] generates ODEs approximating higher-order moments in models using the mass-action kinetics and described in the *Systems Biology Markup Language*. Lapin et al. [131] present *SHAVE*, a tool that approximates a Markov population model with polynomial rates by a hybrid stochastic process, solved by a combination of moment-based techniques and direct solution of the Kolmogorov differential equations.

In Chapter 8 we present *GPA*, a tool implementing the framework developed in this thesis. The main focus of GPA is to provide implementation of ODE analysis and related techniques to Markov population models. Similar to Möbius, GPA supports a number of different formalisms and solution techniques.

Chapter 3

Population Continuous-Time Markov Chains

Key contributions	
Summary of ODE analysis techniques for Markov population model	3.1
PCTMCs as an intermediate layer for process algebra models	3.2.2

This chapter bridges the background theory in Chapter 2 and contributions of this thesis in the following chapters. We summarise a number of existing results considering continuous-time Markov chains (CTMCs) where the state space consists of vectors of non-negative integers. We present these in the form of a framework of *Population Continuous-Time Markov Chains* (PCTMCs). A PCTMC is a CTMC where the states are vectors of numerical populations and the transitions between states can be defined by constant changes in some of the populations, with rates expressible as functions of populations. This definition is common to most of the approaches mentioned in Section 2.2. We show how existing mean-field and higher-order moment ODE analysis techniques apply to PCTMCs. For example, we show that the ODE analysis of models in the GPEPA process algebra by Hayden and Bradley [99] can be interpreted as an application of a moment closure technique to the PCTMC that represents a process algebraic model. Section 3.2.2 presents the semantics of a GPEPA model as an equivalent PCTMC. A similar translation can be defined for other population-based process algebras. This separation of model description and the underlying stochastic process, as shown in Figure 3.1, allows these results to be applied to a range of different formalisms. In this thesis, we present extensions of the PCTMC framework. For example Chapter 5 shows how to capture certain accumulated rewards in the models. This and other extensions are directly applicable to any formalism that can be described by an underlying PCTMC.

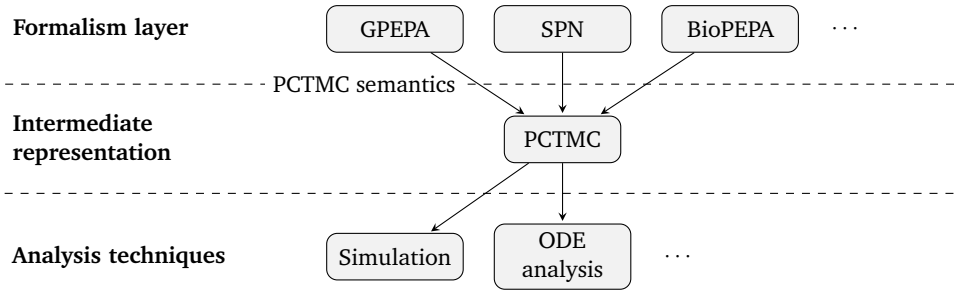


Figure 3.1: PCTMC as an intermediate representation for Markov population formalisms.

3.1 PCTMCs

A *population continuous-time Markov chain* (*PCTMC*) is a CTMC with state space that is a finite vector of non-negative integers, $\boldsymbol{X} = (X_1, \ldots, X_N) \in \mathbb{Z}_+^N$, with the i-th component representing a population of an agent in a state i. The transitions in a PCTMC are described by a set of *transition classes* $\mathcal{C}$. Each transition class $c = (r_c, \boldsymbol{\delta}_c) \in \mathcal{C}$ describes stochastic events in a state $\boldsymbol{X}(t)$ at time t

(i) with exponentially distributed duration D_c at rate $r_c(\boldsymbol{X}(t))$ where $r_c : \mathbb{Z}_+^N \to \mathbb{R}$ is a *rate function* and

(ii) change the current population vector according to the *change vector* $\boldsymbol{\delta}_c$.

The state after each transition from a class c is

$$\boldsymbol{X}(t + D_c) = \boldsymbol{X}(t) + \boldsymbol{\delta}_c.$$

We use PCTMCs to model systems with a large number of interacting agents. The agents can belong to different types and an agent of each type can move between a finite number of distinct states. The populations indexed by $1, \ldots, N$ span all the possible agent – state combinations. We will often index the state space by labels, for example C, S for clients and servers, and assume an implicit bijective mapping between such labels and the set $1, \ldots, N$.

The PCTMC framework is similar to chemical reaction systems, where $\boldsymbol{X}(t)$ describes the counts of different molecules in a chemical solution and transition classes represent chemical reactions between the molecules, where r_c is the reaction propensity function and $\boldsymbol{\delta}_c$ the stoichiometric vector. For clarity, we adapt a notation similar to that of chemical reactions and denote by

$$in_1 + \cdots + in_k \to out_1 + \cdots + out_l \qquad \text{at } r(\boldsymbol{X}) \tag{3.1}$$

where $1 \leq in_i, out_j \leq N$, a transition class c with change vector

$$\boldsymbol{\delta}_c = (\#_1(\boldsymbol{out}) - \#_1(\boldsymbol{in}), \ldots, \#_N(\boldsymbol{out}) - \#_N(\boldsymbol{in})) \in \mathbb{Z}^N$$

where $\#_h(\boldsymbol{v})$ gives the count of h in a vector $\boldsymbol{v}$, and rate

$$r_c(\boldsymbol{X}) = \begin{cases} r(\boldsymbol{X}) & \text{if } X_i \geq \#_i(in) \text{ for all } i = 1, \ldots, N \\ 0 & \text{otherwise} \end{cases}$$

We also allow empty left-hand side or empty right-hand side in Equation 3.1 corresponding to transitions where populations are only increased or decreased respectively.

When defining the rate function $r(\boldsymbol{X})$, we replace the population indexed by a label X_l by the label l itself. For example, if the rate is $r(\boldsymbol{X}) = X_S \cdot X_C$, we write $r(\boldsymbol{X}) = S \cdot C$.

It is appropriate to note here that a PCTMC can represent any CTMC with a finite state space – each state would be assigned a population that can take values 0 and 1 and exactly one population would be non-zero at each time. All the techniques described in this chapter and developed in

this thesis would apply to such a PCTMC. However, in most cases, there would be little or no benefit in the presented analysis techniques compared to directly analysing the CTMC. In general, PCTMCs are suited to models where at least some of the populations could take large values so that the resulting combinatorial explosion would make traditional CTMC analyses too expensive. We show several such models below and later in this thesis.

3.2 Examples

We first present several PCTMC models that will motivate the developments in this thesis. We will show models with different transition rate functions, which influence the available analysis techniques.

3.2.1 Peer-to-peer model

A simple and commonly used transition rate function in PCTMCs is the quadratic *mass-action* kinetics, where the rate function depends on a product of two populations. This assumes that agents interact in a homogeneous environment where they can meet uniformly at random. The rate of interactions depends on the number of possible pairs of agents, captured by the product of respective populations. A common example of such transitions can be found in models of chemical reactions, predator–prey systems or epidemiology models. In performance modelling of computer systems, such rates can be for example found in models of peer-to-peer systems [e.g. 118]. There, the environment represents an internet-wide network and is assumed to be approximately homogeneous. We present a PCTMC describing a simple peer-to-peer system.

The system consists of users who already own a copy of some data to be distributed. Other users are trying to obtain the data. Additionally, to increase the speed of data distribution, the system includes dedicated servers that can perform faster seeding. Users with data can leave the system and potentially return.

The system contains two different agent types – users and dedicated servers. Each user can be in three different states – in possession of the data, without the data and temporarily unavailable, labelled by U_l, U_s, U_f respectively. Each server can be in an on or off state, labelled by S_{on}, S_{off} respectively. The state space consists of vectors of five populations, $\boldsymbol{X} = (U_l, U_s, U_f, S_{on}, S_{off}) \in \mathbb{Z}_+^5$. Initially, there is a fixed number of users and servers in the off state – the initial populations are given by a point mass $(n_{U_l}, n_{U_s}, 0, 0, n_{S_{off}}) \in \mathbb{Z}_+^5$.

We assume that users and servers are uniformly distributed across the network and equally likely to initiate communication with each other. This can be captured by the mass action kinetics – for example, the rate of the event where a user seeds the data to another user without the data is proportional to the product of the two populations, i.e. to $U_l \cdot U_s \cdot r_{seed}$, where r_{seed} is a constant dependent on the data and network properties. Similarly, the rate of seeding from an active server to a user is $S_{on} \cdot U_l \cdot r_{seed,s}$. Additionally, servers can switch between the on and off states and clients can leave and return to the system to obtain the file again. The system behaviour can be captured by six transition classes:

$$S_{off} \to S_{on} \qquad \text{at } S_{off} \cdot r_{on}$$

$$
\begin{aligned}
S_{on} &\to S_{off} && \text{at } S_{on} \cdot r_{off} \\
U_l + U_s &\to U_s + U_s && \text{at } U_l \cdot U_s \cdot r_{seed} \\
U_l + S_{on} &\to U_s + S_{on} && \text{at } U_l \cdot S_{on} \cdot r_{seed,s} \\
U_s &\to U_f && \text{at } U_s \cdot r_{leave} \\
U_f &\to U_l && \text{at } U_f \cdot r_{return}
\end{aligned}
$$

3.2.2 PCTMC semantics of GPEPA

We look at models that can be defined by the GPEPA process algebra. Unlike the peer-to-peer example above, GPEPA assumes *bounded capacity* kinetics for cooperation between a number of agents – the total rate of a cooperation is limited by the slowest agent. We show a simple client–server model, using the syntax described in more detail in Section 2.2.3. We formally define a semantics of GPEPA which translates a GPEPA model into an equivalent PCTMC.

Traditionally, the semantics of GPEPA is defined as a CTMC and further reasoning about the symmetry of the state space can reduce the model into a PCTMC. We define an equivalent semantics of GPEPA directly as a PCTMC. We define three functions on the structure of GPEPA models (described in Section 2.2.3) giving the set of all possible population labels, the set of transition classes and the initial populations in the resulting PCTMC.

Population labels Population labels in the resulting PCTMC correspond to all the possible pairs of group labels and agent states. Function $Ps(M)$ returns the set of all such pairs for a GPEPA model M:

$$
\begin{aligned}
Ps(\mathbf{G}\{P_1[n_1]|\cdots|P_m[n_m]\}) &= \{(\mathbf{G}, P_1), \ldots, (\mathbf{G}, P_m)\} \\
Ps(M_1 \underset{A}{\bowtie} M_2) &= Ps(M_1) \cup Ps(M_2)
\end{aligned}
$$

We assume an implicit bijection between the group – agent pairs in a model M and the set $\{1, \ldots, |Ps(M)|\}$.

Initial populations The initial populations for a model M are $\boldsymbol{x_0} = Init(M)$ where

$$
\begin{aligned}
Init(\mathbf{G}\{P_1[n_1]|\cdots|P_m[n_m]\}) = \{&(\mathbf{G}, P_1) \mapsto n_1, \ldots, (\mathbf{G}, P_m) \mapsto n_m, \\
&(\mathbf{G}, P) \mapsto 0, P \notin \{P_1 \ldots, P_m\}\} \\
Init(M_1 \underset{A}{\bowtie} M_2) &= Init(M_1) \cup Init(M_2)
\end{aligned}
$$

Transition classes Function $Trans(M)$ returns returns the set of all transition classes in the resulting PCTMC. In order to respect synchronisation sets in a GPEPA model, we annotate each transition class with the respective GPEPA action label. For a simple labelled GPEPA group $M = \mathbf{G}\{P_1[n_1]|\cdots|P_m[n_m]\}$, the PCTMC has a transition class for each PEPA transition $P_i \xrightarrow{(\alpha, r)} P_i'$ labelled with an action α. The set $Trans(M)$ consists of

$$
(\mathbf{G}, P_i) \to (\mathbf{G}, P_i') \qquad \text{at rate } X_{(\mathbf{G},P_i)} \cdot r, \textit{ label } \alpha
$$

Let model M be a cooperation of two GPEPA models $M_1 \underset{A}{\bowtie} M_2$ and let $\mathcal{C}_i$ be the sets of labelled transition classes of PCTMCs corresponding to the models M_i respectively. The set of transition classes for the PCTMC of M, $Trans(M)$ consists of

(i) All transition classes in $\mathcal{C}_1$ and $\mathcal{C}_2$ not labelled by an action α in the cooperation set A.

(ii) For each combination of transition classes in $\mathcal{C}_1$ and $\mathcal{C}_2$ labelled by an action α in the cooperation set, that is

$$\sum_{i=1}^{k}(\mathbf{G}_i, P_i) \to \sum_{i=1}^{k'}(\mathbf{G}_i, P_i') \qquad \text{at rate } r_1(\boldsymbol{X}), \text{ label } \alpha$$
$$\sum_{i=1}^{l}(\mathbf{H}_i, Q_i) \to \sum_{i=1}^{l'}(\mathbf{H}_i, Q_i') \qquad \text{at rate } r_2(\boldsymbol{X}), \text{ label } \alpha$$

the transition class defined as:

$$\sum_{i=1}^{k}(\mathbf{G}_i, P_i) + \sum_{i=1}^{l}(\mathbf{H}_i, Q_i) \to \sum_{i=1}^{k'}(\mathbf{G}_i, P_i') + \sum_{i=1}^{l'}(\mathbf{H}_i, Q_i') \quad \text{at rate } r(\boldsymbol{X}), \text{ label } \alpha$$

where the rate is

$$r(\boldsymbol{X}) = \frac{r_1(\boldsymbol{X})}{r_\alpha(M_1, \boldsymbol{X})}\frac{r_2(\boldsymbol{X})}{r_\alpha(M_2, \boldsymbol{X})} \cdot \min\left(r_\alpha(M_1, \boldsymbol{X}), r_\alpha(M_2, \boldsymbol{X})\right)$$

where $r_\alpha(M_i, \boldsymbol{X})$ is the GPEPA apparent rate of the action α in model M_i, defined in Equation 2.6,

$$r_\alpha(M_i, \boldsymbol{X}) = \sum_{(r_c, \boldsymbol{\delta}_c) \in \mathcal{C}_i^\alpha} r_c(\boldsymbol{X})$$

for $\mathcal{C}_i^\alpha$ the set of all transition classes in $\mathcal{C}_i$ labelled with α.

The resulting PCTMC for a GPEPA model M can be constructed as

$$(\boldsymbol{X}(t),\, Trans(M), Init(M))$$

where $\boldsymbol{X}(t)$ is indexed by population labels in $Ps(M)$.

3.2.3 GPEPA client–server model

Using the PCTMC semantics of GPEPA, we can obtain a PCTMC representing the client–server model from Section 2.2.3. There are six group – agent pairs

$$Ps(M) = \big\{(\textbf{Clients}, \mathit{Client}), (\textbf{Clients}, \mathit{Client\_wait}), (\textbf{Clients}, \mathit{Client\_think}),$$
$$(\textbf{Servers}, \mathit{Server}), (\textbf{Servers}, \mathit{Server\_get}), (\textbf{Servers}, \mathit{Server\_broken})\big\}$$

We will abbreviate the individual labels and also population vectors, and write each state at time t as $(C(t), C_w(t), C_t(t), S(t), S_g(t), S_b(t)) \in \mathbb{Z}_+^6$. The initial state given by the system equation is $Init(M) = (n_C, 0, 0, n_S, 0, 0)$.

The construction above gives five transition classes in total:

$$
\begin{aligned}
C + S &\to C_w + S_g && \text{at } r_{request} \cdot \min(C, S) \\
C_w + S_g &\to C_t + S && \text{at } r_{data} \cdot \min(C_w, S_g) \\
C_t &\to C && \text{at } r_{think} \cdot C_t \\
S &\to S_b && \text{at } r_{break} \cdot S \\
S_b &\to S && \text{at } r_{reset} \cdot S_b
\end{aligned}
$$

3.3 Simulation

A straightforward way to analyse a PCTMC is to numerically compute individual realisations of the stochastic process. Given a PCTMC $(\boldsymbol{X}(t), \mathcal{C}, \boldsymbol{X_0})$ and a finite time $t_f \in \mathbb{R}$, the following algorithm, often attributed to Gillespie [82], calculates a trace $\boldsymbol{x}(t)$ for $0 \leq t \leq t_f$ of the PCTMC:

1. Sample an initial state $\boldsymbol{x}(0)$ according to the distribution given by $\boldsymbol{X_0}$. Set $t = 0$.
2. For the current state at time t, $\boldsymbol{x}(t)$, calculate the values of rate functions $r_c(\boldsymbol{x}(t))$ for each $c \in \mathcal{C}$. Calculate the sum of all rates $r = \sum_{c \in \mathcal{C}} r_c(\boldsymbol{x}(t))$.
3. Sample the time τ until the next transition, from an exponential distribution with parameter r.
4. Sample the next transition class i, from discrete distribution with $\mathbb{P}(i = c) = r_c(\boldsymbol{x}(t))/r$ for each $c \in \mathcal{C}$.
5. Set $\boldsymbol{x}(t + s) = \boldsymbol{x}(t)$ for $s < \tau$ and set the new state $\boldsymbol{x}(t + \tau) = \boldsymbol{x}(t) + \boldsymbol{\delta}_i$ and $t = t + \tau$.
6. Repeat steps 2 to 5 until $t \geq t_f$.

We can repeatedly apply this algorithm to compute a large number of traces. These can be used to obtain estimates of statistical properties of the stochastic process, such as the distribution or moments of the population vector $\boldsymbol{X}(t)$ at some time $t \leq t_f$.

The computational complexity of this approach depends on the size of the state space, the time t_f and the particular form of the rate functions. In many cases, we will look at systems where the rates linearly depend on the total number of agents. Scaling such systems by a constant factor S decreases the number of traces required for statistically significant results. At the same time, the frequency of transitions in the PCTMC increases roughly by a factor of S. Overall, simulation of PCTMC models can be expensive for even a single parameter configuration in the model.

3.4 ODE analysis of PCTMCs

Simulation of PCTMCs and traditional explicit state space analysis techniques of CTMCs from Section 2.1.1 become costly when applied to large PCTMC models. Traditional methods have to keep track of the stochastic dependencies between individual populations by explicitly calculating

probabilities for all possible combinations of values of populations. This results in state space explosion even if sophisticated aggregation schemes are used. On the other hand, stochastic simulation evaluates sample paths of the process and the stochastic dependencies are only captured by calculating statistical estimates from a large number of independently computed paths.

In this section, we present several results concerning the so-called *ODE analysis* of PCTMCs. The ODE analysis derives a deterministic, real-valued process that approximates the evolution of populations over time. For example, the *mean-field* analysis defines a set of ordinary differential equations (ODEs) that approximate the evolution of average values of populations over time. Dependencies between populations are captured by simultaneously solving the set of ODEs for all populations, ignoring any covariance between the individual population variables. This greatly reduces the cost of the analysis, now not depending on initial populations in the model and not suffering from high transition rates. Although this approximation is usually accurate, it can lead to high errors, sometimes preventing the modeller from gaining a good understanding of the evolution of the system. It is possible to reduce the error by accounting for covariance between populations. The ODE system can be extended with auxiliary variables capturing higher-order moments of populations. Usually, further approximations, referred to as *moment closures*, have to be included to make the resulting system of ODEs finite. We give an overview of several moment closures in Section 3.4.2 below and present a novel moment closure for models with minimum rates in Section 4.4.

The following theorem gives an exact form of the ODEs describing a moment of populations in a PCTMC [e.g. 95, 71], and serves as the basis of different variants of ODE analysis:

Theorem 1. For a PCTMC $(\boldsymbol{X}(t), \mathcal{C}, \boldsymbol{X_0})$ and a moment function $h\colon \mathbb{R}_+^N \to \mathbb{R}$, the moment described by h evolves according to a differential equation

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[h(\boldsymbol{X}(t))] = \mathbb{E}[f_h(\boldsymbol{X}(t))] \tag{3.2}$$

where

$$f_h(\boldsymbol{X}(t)) = \sum_{c\in\mathcal{C}} (h(\boldsymbol{X}(t) + \boldsymbol{\delta}_c) - h(\boldsymbol{X}(t)))r_c(\boldsymbol{X}(t)) \tag{3.3}$$

For example, to obtain the ODE describing the evolution of a mean of a population $X_i(t)$, we set $h(\boldsymbol{X}) = X_i$ and get

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[X_i(t)] = \mathbb{E}\left[\sum_{(\boldsymbol{\delta}_c, r_c)\in\mathcal{C}} \delta_{cj} r_c(\boldsymbol{X}(t))\right] \tag{3.4}$$

Similarly, for higher-order moments we use a suitable moment function $h(\boldsymbol{X})$, for example $h(\boldsymbol{X}) = X_i X_j$ for the evolution of the mean of the product of populations of i and j.

Some terms on the right-hand side of an ODE from Theorem 1 do not necessarily have a known closed form. In some cases, the expectation $\mathbb{E}[f_h(\boldsymbol{X}(t))]$ is a linear combination of additional

moments that can be captured by Theorem 1. However, in most cases, an approximation has to be applied in order to replace the expectation by an expression involving only moments where Theorem 1 applies. The following two cases of ODE analysis apply such approximations to construct a closed system of ODEs that captures moments of populations in PCTMCs.

3.4.1 Mean-field analysis

Mean-field analysis constructs a system of ODEs with a solution $\boldsymbol{x}(t) \in \mathbb{R}_+^N$, where each element of $\boldsymbol{x}(t)$ corresponds to a population mean. The ODEs are obtained by applying Theorem 1 to population means and by approximating the expectation of a rate function by evaluating the function on expectations of its arguments:

$$\mathbb{E}[r(\boldsymbol{X}(t))] \approx r(\mathbb{E}[\boldsymbol{X}(t)]) \tag{3.5}$$

This gives a closed system of ODEs

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t) &= \boldsymbol{f}_{\boldsymbol{X}}(\boldsymbol{x}(t)) \\ \boldsymbol{x}(0) &= \mathbb{E}[\boldsymbol{X}_{\boldsymbol{0}}] \end{aligned} \tag{3.6}$$

where

$$\boldsymbol{f}_{\boldsymbol{X}}(\boldsymbol{x}(t)) = \begin{pmatrix} f_{h_1}(\boldsymbol{x}(t)) \\ \cdots \\ f_{h_N}(\boldsymbol{x}(t)) \end{pmatrix} \tag{3.7}$$

for $h_i(\boldsymbol{X}(t)) = X_i(t)$ and f_{h_i} from Theorem 1.

In Section 3.6, we present a result that shows convergence between the mean-field approximation and the means in a PCTMC. Therefore we adopt the following notation:

$$\begin{aligned} \widetilde{\mathbb{E}}[\boldsymbol{X}(t)] &\stackrel{def}{=} \boldsymbol{x}(t) \\ \widetilde{\mathbb{E}}[X_i(t)] &\stackrel{def}{=} x_i(t) \qquad\qquad 1 \le i \le N \end{aligned} \tag{3.8}$$

where $X_i(t)$ can be replaced by the respective population label shorthand, such as in $\widetilde{\mathbb{E}}[C(t)]$ for the approximation of mean client population in the client–server model. The general structure of mean-field equations can be seen in Figure 3.2.

$$\begin{aligned} \tfrac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[X_i(t)] \;=\; & \mathbb{E}[f(X_1(t), \ldots, X_n(t))] \qquad \textbf{(First order)} \\ & \approx \\ & f(\,\widetilde{\mathbb{E}}[X_1(t)], \ldots, \widetilde{\mathbb{E}}[X_n(t)]) \end{aligned}$$

Figure 3.2: Structure of mean-field ODEs.

In the client–server model, the equations are

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[C(t)] &= -\min(\widetilde{\mathbb{E}}[S(t)],\widetilde{\mathbb{E}}[C(t)])\cdot r_{request} + \widetilde{\mathbb{E}}[C_t(t)]\cdot r_{think}\\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[C_w(t)] &= -\min(\widetilde{\mathbb{E}}[C_w(t)],\widetilde{\mathbb{E}}[S_g(t)])\cdot r_{data} + \min(\widetilde{\mathbb{E}}[S(t)],\widetilde{\mathbb{E}}[C(t)])\cdot r_{request}\\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[C_t(t)] &= -\widetilde{\mathbb{E}}[C_t(t)]\cdot r_{think} + \min(\widetilde{\mathbb{E}}[C_w(t)],\widetilde{\mathbb{E}}[S_g(t)])\cdot r_{data}\\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[S(t)] &= -\min(\widetilde{\mathbb{E}}[S(t)],\widetilde{\mathbb{E}}[C(t)])\cdot r_{request} - \widetilde{\mathbb{E}}[S(t)]\cdot r_{break}\\
&\quad + \min(\widetilde{\mathbb{E}}[C_w(t)],\widetilde{\mathbb{E}}[S_g(t)])\cdot r_{data} + \widetilde{\mathbb{E}}[S_b(t)]\cdot r_{reset}\\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[S_g(t)] &= -\min(\widetilde{\mathbb{E}}[C_w(t)],\widetilde{\mathbb{E}}[S_g(t)])\cdot r_{data} + \min(\widetilde{\mathbb{E}}[S(t)],\widetilde{\mathbb{E}}[C(t)])\cdot r_{request}\\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[S_b(t)] &= -\widetilde{\mathbb{E}}[S_b(t)]\cdot r_{reset} + \widetilde{\mathbb{E}}[S(t)]\cdot r_{break}
\end{aligned}$$

with initial conditions

$$\widetilde{\mathbb{E}}[\boldsymbol{X}(0)] = (n_C, 0, 0, n_S, 0, 0)^{\mathrm{T}}$$

3.4.2 Moment closures

Theorem 1 can be also used to obtain ODEs that describe the evolution of higher-order moments of populations. Structure of the resulting system of ODEs and the exact form of the right-hand sides depend on rates used in the PCTMC transition classes.

Moment closure for GPEPA models

Transition rate functions $r\colon \mathbb{R}_+^N \to \mathbb{R}$ for GPEPA models enjoy the following homogeneity property [99]: for all $y \in \mathbb{R}_+$ and $\boldsymbol{x} \in \mathbb{R}_+^N$,

$$y \cdot r(\boldsymbol{x}) = r(y \cdot \boldsymbol{x}). \tag{3.9}$$

This permits Equation 3.5 to be applied to right-hand sides of ODEs describing raw moments of populations in a PCTMC $\boldsymbol{X}(t)$ that is obtained from a GPEPA model, and use the following approximation:

$$\mathbb{E}[Y \cdot r(\boldsymbol{X}(t))] = \mathbb{E}[r(Y \cdot \boldsymbol{X}(t))] \approx r(\widetilde{\mathbb{E}}[Y \cdot \boldsymbol{X}(t)]) \tag{3.10}$$

where Y is a non-negative random variable, such as a product of populations $X_{i_1}(t)\cdots X_{i_k}(t)$.

This approximation can be directly applied to right-hand sides of ODEs for higher-order moments of populations. For a raw moment of populations of order o, given by $h(\boldsymbol{X}) = X_{i_1}\cdots X_{i_o}$, the difference $h(\boldsymbol{X}(t)+\boldsymbol{\delta}_c) - h(\boldsymbol{X}(t))$ on the right-hand side of the ODE for $\mathbb{E}[h(\boldsymbol{X}(t))]$ consists of terms which are products of up to $(o-1)$ populations. By definition of the GPEPA semantics, the rate $r(\boldsymbol{X}(t))$ does not contain terms with any products of populations. Therefore, after applying the approximation from Equation 3.10, the right-hand side of the approximate moment ODE for $\mathbb{E}[h(\boldsymbol{X}(t))]$ is an expression composed of moments of order up to o and using only addition, scalar

multiplication, applications of the minimum function and the PEPA division operator. We can apply Theorem 1 and the approximation from Equation 3.10 to all raw moments of populations of order up to and including o and obtain a closed system of ODEs.

For example, if we set $o = 2$, we get $\boldsymbol{x}(t) = (x_1(t), \ldots, x_N(t), x_{\{1,1\}}(t), x_{\{1,2\}}(t), \ldots, x_{\{N,N\}}(t))^{\mathrm{T}}$ where $x_i(t)$ approximates the mean of population i and $x_{\{i,j\}}(t)$ the joint moment $\mathbb{E}[X_i(t)X_j(t)]$. We get a closed system of $N + N \cdot (N+1)/2$ ODEs:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t) &= \boldsymbol{f}_{\boldsymbol{X}^2}(\boldsymbol{x}(t)) \\ \boldsymbol{x}(0) &= (\mathbb{E}[\boldsymbol{X_0}]^{\mathrm{T}}, \mathbb{E}[X_{0,i} \cdot X_{0,j}]\ 1 \leq i \leq j \leq N)^{\mathrm{T}} \end{aligned} \tag{3.11}$$

where

$$f_{\boldsymbol{X}^2}(\boldsymbol{x}(t)) = \begin{pmatrix} f_{h_1}(\boldsymbol{x}(t)) \\ \cdots \\ f_{h_N}(\boldsymbol{x}(t)) \\ g_{h_{\{1,1\}}}(\boldsymbol{x}(t)) \\ g_{h_{\{1,2\}}}(\boldsymbol{x}(t)) \\ \cdots \\ g_{h_{\{N,N\}}}(\boldsymbol{x}(t)) \end{pmatrix} \tag{3.12}$$

for $h_i(\boldsymbol{X}) = X_i$, $h_{\{i,j\}}(\boldsymbol{X}) = X_i \cdot X_j$. Functions f_{h_i} are derived directly from Theorem 1. Functions $g_{h_{\{i,j\}}}$ are obtained by applying the approximation from Equation 3.10 to the respective functions $f_{h_{\{i,j\}}}$ from Theorem 1 where occurrences of second-order moments $\mathbb{E}[X_k(t)X_l(t)]$ are replaced by the corresponding elements $x_{\{k,l\}}(t)$ from $\boldsymbol{x}(t)$.

A similar set of ODEs can be obtained for higher-order moments of populations. We extend the notation from Equation 3.8 and write

$$\widetilde{\mathbb{E}}[X_{i_1}(t) \cdots X_{i_k}(t)] \stackrel{def}{=} x_{i_1,\ldots,i_k}(t) \qquad 1 \leq i_1, \ldots, i_k \leq N \tag{3.13}$$

> In the client–server model, there are six different populations and 21 possible second-order moments, giving the state space $\boldsymbol{X}(t) \in \mathbb{Z}_+^{27}$. The approximation above gives the differential equation for the moment $\widetilde{\mathbb{E}}[C(t)S(t)]$:
>
> $$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[C(t)S(t)] = &- \min(\widetilde{\mathbb{E}}[C(t)S(t)], \widetilde{\mathbb{E}}[S(t)^2]) \cdot r_{request} - \min(\widetilde{\mathbb{E}}[C(t)^2], \widetilde{\mathbb{E}}[S(t)C(t)]) \cdot r_{request} \\ &+ \min(\widetilde{\mathbb{E}}[C(t)], \widetilde{\mathbb{E}}[S(t)]) \cdot r_{request} + \widetilde{\mathbb{E}}[C_t(t)S(t)] \cdot r_{think} \\ &+ \widetilde{\mathbb{E}}[S_b(t)C(t)] \cdot r_{reset} - \widetilde{\mathbb{E}}[S(t)C(t)] \cdot r_{break} + \min(\widetilde{\mathbb{E}}[C_w(t)C(t)], \widetilde{\mathbb{E}}[S_g(t)C(t)]) \cdot r_{data} \end{aligned}$$
>
> The complete system of 27 ODEs can be found in Appendix A.1.2.

The general structure of moment ODEs for a GPEPA based PCTMC can be seen in Figure 3.3. Note that an ODE for a moment does not depend on an ODE for a higher moment. In particular, the resulting approximation for means still ignores any covariances between populations. In

Section 4.4 we present a new moment closure which introduces this dependency and leads to more accurate approximations.

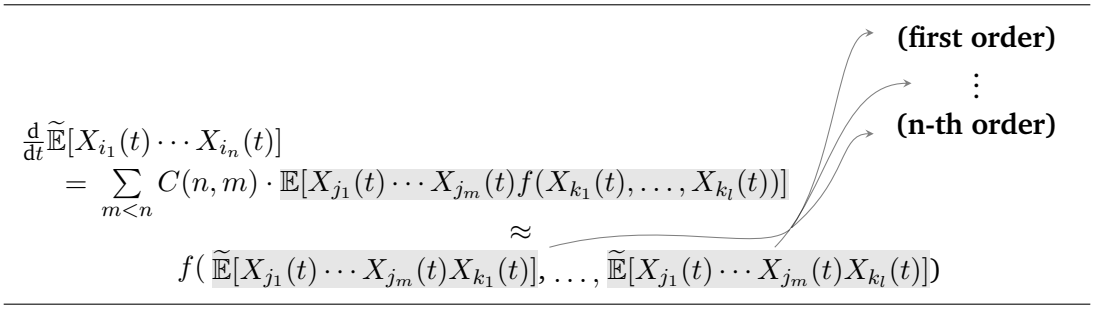


Figure 3.3: Structure of higher-order moment GPEPA ODEs.

3.4.3 Normal closure for PCTMCs with polynomial rates

For a large class of PCTMC models including the peer-to-peer model in Section 3.2.1, the rates of transitions are defined as polynomial functions of populations: $r(\boldsymbol{X}(t)) = \sum_{i=0}^{m} c_i \prod_{j=1}^{i} X_{k_j}(t)$ for an order $m \in \mathbb{Z}_+$ and constants $c_i \in \mathbb{R}$, $0 \leq i \leq m$. This rate occurs on the right-hand side of an ODE for a moment of order o, given by $h(\boldsymbol{X}) = X_{i_1} \cdots X_{i_o}$, and is multiplied by terms of the form $X_{j_1}(t) \cdots X_{j_l}(t)$ where $l < o$ (these originate from the difference $h(\boldsymbol{X}(t) + \delta_c) - h(\boldsymbol{X}(t))$ for a transition class c). Therefore the right-hand side of the ODE for $\mathbb{E}[h(\boldsymbol{X}(t))]$ is composed of moments of orders up to and including $o - 1 + m$. If $m > 1$, the moment ODE depends on moments of higher orders. Although it is possible to apply Theorem 1 to those moments, doing so would result in an infinite system of ODEs – each moment would depend on a moment of a higher order. In the peer-to-peer model, the ODE for the moment $\mathbb{E}[U_l(t)]$ would contain the moment $\mathbb{E}[U_l(t)\,U_s(t)]$, the ODE for $\mathbb{E}[U_l(t)\,U_s(t)]$ the moment $\mathbb{E}[U_l(t)^2\,U_s(t)]$ and so on.

One way to reduce this system to a finite number of equations is to apply an approximation that transforms the right-hand side of an ODE for a moment of order o in a way that only moments of order up to o are included. A trivial case of this approximation is used in the mean-field approach, where for example the term $\mathbb{E}[U_l(t)\,U_s(t)]$ is reduced to $\mathbb{E}[U_l(t)] \cdot \mathbb{E}[U_s(t)]$, ignoring the covariance between $U_l(t)$ and $U_s(t)$.

There are infinitely many possibilities for such an approximation of higher-order moments. Often, an approximation is chosen according to an assumption about the distribution of the populations. Probably the most common choice is the normal distribution. This is further justified by a convergence theorem presented in Section 3.6.2 which empirically shows that the normal distribution is a suitable choice as the scale of the system increases. If we assume that the population vector $\boldsymbol{X}(t)$ is a multivariate normal random variable at each time t, we can use a known theorem which expresses central moments of a multivariate normal random variable as a linear combination of second-order moments:

Theorem 2 (Isserlis' theorem [113])**.** Let $\boldsymbol{X} \in \mathbb{R}^N$ be a multivariate normal variable with mean $\boldsymbol{\mu}$ and covariance matrix (σ_{ij}). Every joint central moment of an order greater than two, given by $\mathbb{E}[(X_1 - \mu_1)^{m_1} \cdots (X_N - \mu_N)^{m_N}]$ for $\boldsymbol{m} \in \mathbb{Z}_+^N$, can be expressed as a multinomial of second-order

moments:

$$\begin{aligned}\mathbb{E}[(X_1-\mu_1)^{m_1}\cdots(X_N-\mu_N)^{m_N}] &= 0, && \text{if } \textstyle\sum m_i \text{ is odd}\\ \mathbb{E}[(X_1-\mu_1)^{m_1}\cdots(X_N-\mu_N)^{m_N}] &= \textstyle\sum\prod \mathbb{E}[(X_i-\mu_i)(X_j-\mu_j)] && \text{if } \textstyle\sum m_i \text{ is even}\end{aligned} \tag{3.14}$$

where $\sum\prod$ sums through all the distinct partitions of $1,\ldots,N$ into disjoint sets of pairs (i,j).

We use this theorem to obtain an approximation to a raw moment $\mathbb{E}[X_1(t)^{m_1}\cdots X_N(t)^{m_N}]$. We simply expand the central moment in Equation 3.14 and subsequently re-arrange the equation to obtain an approximation involving moments of order less than $\sum m_i$.

> In the peer-to-peer model, instead of including an ODE for the third-order joint raw moment $\mathbb{E}[U_s(t)\,U_l(t)^2]$ we can close the expansion by using the approximation
>
> $$\begin{aligned}\mathbb{E}\left[(U_s(t)-\mathbb{E}[U_s(t)])(U_l(t)-\mathbb{E}[U_l])^2\right] &\approx 0\\ \mathbb{E}\left[U_s(t)\,U_l(t)^2\right] &\approx 2\mathbb{E}[U_l(t)]\mathbb{E}[U_l(t)\,U_s(t)] + \mathbb{E}[U_s(t)]\mathbb{E}[U_l(t)^2]\\ &\quad - 2\mathbb{E}[U_s(t)]\mathbb{E}[U_l(t)]^2\end{aligned}$$
>
> We have a choice at which order to apply this approximation. For example, instead of approximating the moment $\mathbb{E}[U_l(t)^2\,U_s(t)]$, we can include the ODE for this moment and approximate the fourth-order moments on its right-hand side with moments of order one, two and three.

In general, we get a family of *moment closure* approximations. We can choose a maximal order o_m and apply the above approximation to all moments of order above o_m. This leads to a closed system of ODEs with solution approximating all moments of orders up to and including o_m. The solution vector $\boldsymbol{x}(t)$ is indexed by the moments – each can be represented as a multiset of population indices $1,\ldots,N$:

$$\begin{aligned}\boldsymbol{x}(t) &= (x_1(t),\ldots,x_N(t), x_{\{i_1,\ldots,i_k\}}(t), k\le o_m, 1\le i_j\le N)^{\mathrm{T}}\\ \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t) &= \boldsymbol{f}_{\boldsymbol{X},N(o_m)}(\boldsymbol{x}(t))\\ \boldsymbol{x}(0) &= (\mathbb{E}[\boldsymbol{X_0}], \mathbb{E}[X_{0,i_1}\cdots X_{0,i_k}], k\le o_m, 1\le i_j\le N)^{\mathrm{T}}\end{aligned}$$

where

$$f_{\boldsymbol{X},N(o_m)}(\boldsymbol{x}(t)) = \begin{pmatrix} g_{h_1}(\boldsymbol{x}(t))\\ \cdots\\ g_{h_N}(\boldsymbol{x}(t))\\ g_{h_{\{i_1,\ldots,i_k\}}}(\boldsymbol{x}(t)), k\le o_m, 1\le i_j\le N\end{pmatrix} \tag{3.15}$$

for $h_i(\boldsymbol{X}) = X_i$, $h_{\{i_1,\ldots,i_k\}}(\boldsymbol{X}) = X_{i_1}\cdots X_{i_k}$. The functions g_h are derived from the respective functions f_h from Theorem 1, with occurrences of moments of order above o_m replaced by the expression given by Theorem 2 and occurrences of moments $\mathbb{E}[X_{i_1}(t)\cdots X_{i_k}(t)]$ replaced by $x_{\{i_1,\ldots,i_k\}}(t)$. Figure 3.4 shows the general structure of the resulting ODE system for a model with quadratic rates. Each ODE for a moment of order k depends on moments of orders up to and

including $k+1$. The normal approximation is applied at the order $n = o_m$ where each moment of order $n+1$ gets replaced by an expression consisting moments of orders up to n.

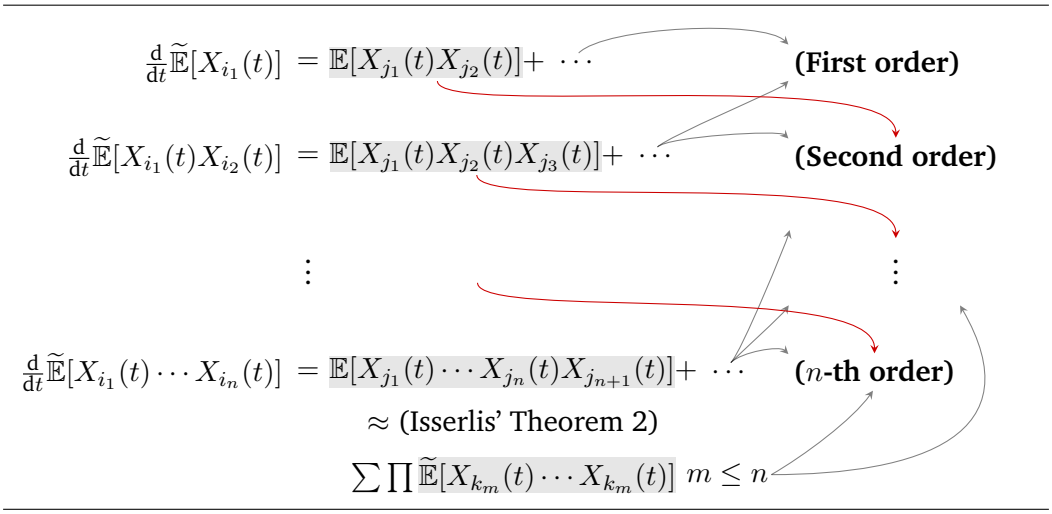


Figure 3.4: Structure of moment ODEs for a PCTMC with quadratic rates closed at order n.

3.4.4 Numerical solutions of mean-field and moment ODEs

Both the mean-field and moment closure methods produce a system of ODEs which approximate the transient evolution of a PCTMC. In most practical situations, the system of ODEs is non-linear without a known closed-form solution. In order to solve such system for a finite interval of time, it is often necessary to resort to *numerical integration methods*.

The most basic such method is the Euler method. Given an initial value for the solution to the ODEs and a fixed *step size* h, the method proceeds to numerically evaluate the solution at times equal to multiples of h. The value of h provides a trade-off between computational cost and accuracy. Euler method, which is rarely used in practice, is the simplest in the family of Runge-Kutta methods [e.g. 46]. Most of the examples in the remainder of this thesis were produced using the fourth order Runge-Kutta method. It was usually sufficient to select step size to be one tenth of the distance between two successive data points on a plot and then manually inspect the solution as compared to stochastic simulation. A more general approach is to use a method which automatically controls the step size based on a current estimate of the error in the solution, such as the algorithm by Dormand and Prince [70]. We used this method when the system of ODEs formed a part of a larger optimisation problem, such as in Section 5.5. The GPA tool provides a number of further methods implemented by the *Apache Commons Mathematics* library[61].

Usually, the computational cost of solving the ODEs is significantly smaller than that of stochastic simulation from Section 3.3. In some cases, for example when there are transition rates at very different orders of magnitude, the resulting system of ODEs is *stiff* [46] and requires a step size which would make the numerical solution prohibitively expensive. We encountered several such instances when solving moment closure ODEs at high orders, seventh and above and were still able to use a suitably small step size to obtain a solution. In future, we are planning to integrate a number of specialised solvers to deal with stiff ODEs.

3.5 Efficient computation of passage times

Knowing the distribution of populations over time or its moments can give a good understanding of the transient evolution of a PCTMC. However, in many practical applications, system providers are primarily interested in a number of derived metrics. Among these are so-called *passage times*. These are random variables which measure the elapsed time of a specified sequence of events. For example, in the client–server model, the server provider can be interested in predicting the time a single client takes to obtain and process its data. Traditionally, computing the distribution or moments of such time variables suffers from the same state-space explosion problems as mentioned in Section 3.4. As reviewed in Section 2.3.1, it is possible to use the ODE analysis to compute passage time metrics in PCTMCs. Unlike other approaches computing only the mean passage time using Little's Law [68, 181], the method of Hayden et al. [4] extends the ODE analysis of GPEPA models to compute probability distributions of a range of passage-time measures. In this section we show a straightforward way to translate this technique to any PCTMC which can be related to a transition graph of individual agent states, and give an overview of the types of passage-time measures the ODE analysis of PCTMCs can compute. Later in this thesis, we will demonstrate the technique on several examples. Most importantly, passage times will play a central role in the performance–energy trade-off problem in Section 5.5.

3.5.1 Agent state transition graphs

In this thesis, we will be interested in passage times derived from the behaviour of individual agents, for example clients and servers in the client–server model. A PCTMC does not represent the system at an agent level. The population structure is flat and there is no relationship between the decreasing and increasing populations in a transition class. For example, in the client–server model, the transition with a change vector represented by $C + S \to C_w + S_g$ does not specify that an agent in the $Client$ state moves to the $Client\_wait$ state; it only describes the changes in respective populations. This additional information is given by the definition of PEPA agents, resulting in a state transition graph, where each connected component corresponds to the state space of each agent. For the client–server model this graph can be seen in Figure 2.3. Whenever applying the passage-time analysis from this section, we require that the PCTMC is augmented with a similar state transition graph for all the agents of interest. This graph has to agree with the definition of transition classes: for a transition with a change vector represented by $s_1 + \cdots + s_k \to t_1 + \cdots + t_l$, there has to be a unique pairing (s_i, t_j), $1 \leq i \leq k$, $1 \leq j \leq l$, such that there is an edge from s_i to t_j in the state graph. For example, in the peer-to-peer model, an obvious choice for the state graph would consist of the edges $U_l \to U_s$, $U_s \to U_f$, $U_f \to U_l$ and $S_{off} \to S_{on}$, $S_{on} \to S_{off}$.

We define passage times on this graph, which now fully describes the behaviour of individual agents. In the peer-to-peer model, we can be interested in the time it takes a single user to obtain the shared data for the first time. In order to apply the techniques described below, we need to specify the passage time as a set of target states $\mathbb{T}$. This set has to be absorbing, that is the set of reachable states from $\mathbb{T}$ needs to be a subset of $\mathbb{T}$. In case the desired set of states is not absorbing, it is possible to modify the state graph to obtain an equivalent model where the set $\mathbb{T}$

is absorbing. We can create an additional type of the user agent and remember whether a data transition has taken place. Agent states U_l*, U_s*, U_f* correspond to the respective states without the $*$ symbol, after the user agent already obtained the data. The resulting model becomes:

$$
\begin{array}{rlrl}
S_{off} \to S_{on} & \text{at } S_{off} \cdot r_{on} & & \\
S_{on} \to S_{off} & \text{at } S_{on} \cdot r_{off} & & \\
U_l + U_s \to U_s* + U_s & \text{at } U_l \cdot U_s \cdot r_{seed} & U_l + U_s* \to U_s* + U_s* & \text{at } U_l \cdot U_s* \cdot r_{seed} \\
U_l* + U_s \to U_s* + U_s & \text{at } U_l* \cdot U_s \cdot r_{seed} & U_l* + U_s* \to U_s* + U_s* & \text{at } U_l* \cdot U_s* \cdot r_{seed} \\
U_l + S_{on} \to U_s* + S_{on} & \text{at } U_l \cdot S_{on} \cdot r_{seed,s} & & \\
U_s \to U_f & \text{at } U_s \cdot r_{leave} & U_s* \to U_f* & \text{at } U_s* \cdot r_{leave} \\
U_f \to U_l & \text{at } U_s \cdot r_{return} & U_f* \to U_l* & \text{at } U_f* \cdot r_{return}
\end{array}
$$

The set $\mathbb{T} = \{U_l*, U_s*, U_f*\}$ is absorbing and can be used in the techniques described below.

In the client–server model, the passage time for a single client agent to obtain and process the data can be defined as the time the client takes to complete a full cycle of transitions from the state *Client* back to itself. To obtain an absorbing set of target states, we can create an additional copy of the client state space, with states *Client*$*$, *Client_wait*$*$ and *Client_think*$*$, which the client enters after reaching the state *Client* for the first time, Figure 3.5.

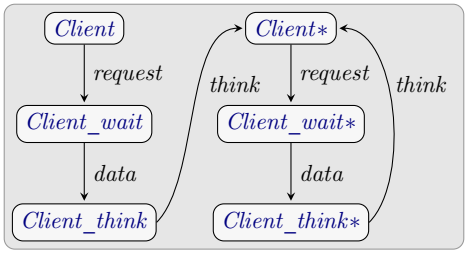


Figure 3.5: Unfolded client state graph with a new, absorbing, set of states.

After expanding the state space, the set of transition classes has to be extended with copies of classes that involve the agents with modified state space. Additionally, rates of existing transition classes have to be modified to account for the competition between agents in the added states and the original states. The exact form of the new rates depends on the modelling formalism. GPEPA naturally deals with such cases: if the same action is offered by multiple agents in one group, the transition rates are defined as weighted proportions of the total synchronisation rate. Moreover, in GPEPA the expanded agent states can be automatically derived from an extended GPEPA agent and the resulting rates can be obtained by applying GPEPA semantics to the new model. We illustrate this construction in the following section.

3.5.2 Probed client–server model

We can define the expanded state space of a client agent in Figure 3.5 directly in GPEPA syntax. We define a GPEPA agent that "remembers" whether a *think* action has fired:

$$NotDone \stackrel{def}{=} (think, r_{think}).Done$$

$$Done \stackrel{def}{=} (think, r_{think}).Done$$

By composing this agent with a single client agent, synchronising on the $think$ action, we obtain a GPEPA agent with a state graph equivalent to that of Figure 3.5.

We can modify the client group, replacing a single client agent with the new tagged version:

$$\mathbf{Clients}\{Client[n_C - 1] | (Client \underset{think}{\bowtie} NotDone)\} \tag{3.16}$$

The Unified Stochastic Probes formalism [103] generalises this approach and is able to automatically generate the attached agent, referred to as a *probe*, from a high level behavioural description of the passage time variable. We will name this version of the client–server model the *probed* client–server model.

The resulting PCTMC now contains six new populations which we label as

$$C_j^D \text{ for } Client_j \underset{think}{\bowtie} Done \text{ and } C_j^{ND} \text{ for } Client_j \underset{think}{\bowtie} NotDone.$$

We can apply GPEPA semantics directly to obtain the new PCTMC. In addition to transition classes involving only server agents, there are nine new transition classes, three for each different copy of the client agent:

$$\begin{aligned}
C^i + S &\to {C_w}^i + S_g && \text{at } r_{request} \cdot \frac{C^i}{r_C(\boldsymbol{X})} \cdot \min(r_C(\boldsymbol{X}), S) \\
{C_w}^i + S_g &\to {C_t}^i + S && \text{at } r_{data} \cdot \frac{{C_w}^i}{r_{C_w}(\boldsymbol{X})} \cdot \min(r_{C_w}(\boldsymbol{X}), S_g) \\
{C_t}^i &\to C^i && \text{at } r_{think} \cdot {C_t}^i
\end{aligned}$$

where $C^i \in \{C, C^D, C^{ND}\}$, ${C_w}^i \in \{C_w, {C_w}^D, {C_w}^{ND}\}$, and

$$\begin{aligned}
r_C(\boldsymbol{X}) &= C + C^{ND} + C^D, \\
r_{C_w}(\boldsymbol{X}) &= C_w + {C_w}^{ND} + {C_w}^D.
\end{aligned}$$

3.5.3 Individual passage time

We are given a single agent with state $A(t) \in \mathbb{S}_A$ over time, where $\mathbb{S}_A$ is an expanded state space containing a target set of absorbing states $\mathbb{T} \subseteq \mathbb{S}_A$. We can define an *individual passage time* random variable T as the first time the agent reaches a state in $\mathbb{T}$, $T \stackrel{def}{=} \inf\{t \in \mathbb{R}_+ \mid A(t) \in \mathbb{T}\}$. The following identity holds because the target state set is absorbing, and can be used to express the CDF of T in terms of expected value of the state process $A(t)$:

$$\mathbb{P}(T \leq t) = \mathbb{P}(A(t) \in \mathbb{T}) = \sum_{i \in \mathbb{T}} \mathbb{P}(A(t) = i) = \sum_{i \in \mathbb{T}} \mathbb{E}[\mathbf{1}_{A(t)=i}] \tag{3.17}$$

In a PCTMC setting, we can substitute one instance of an agent of interest with a copy uniquely labelled with states in $\mathbb{S}_A$, containing an absorbing set of target states $\mathbb{T}$. For example, in Equation 3.16, an instance of the $Client$ agent is replaced with a tagged version $Client \underset{think}{\bowtie} NotDone$.

Then, the right-hand side of Equation 3.17 can be evaluated using population means in the PCTMC: if the resulting PCTMC is $\boldsymbol{X}(t)$, we have

$$\mathbb{P}(T \leq t) = \sum_{i \in \mathbb{T}} \mathbb{E}[X_i(t)].$$

We can use the ODE analysis from Section 3.4 to evaluate this expression, using approximations to mean populations $\widetilde{\mathbb{E}}[\boldsymbol{X}(t)]$. For example, in the client–server model, we would get

$$\mathbb{P}(T \leq t) = \widetilde{\mathbb{E}}\left[C^D(t)\right] + \widetilde{\mathbb{E}}\left[{C_w}^D(t)\right] + \widetilde{\mathbb{E}}\left[{C_t}^D(t)\right]. \tag{3.18}$$

3.5.4 Global passage time

For the so-called global passage times, we are given a PCTMC $\boldsymbol{X}(t)$ with population labels containing agent states with a target absorbing subset $\mathbb{T}$, target population value $p \in \mathbb{Z}_+$ and $n \in \mathbb{Z}_+$, the total invariant population of agents capable of reaching a state in $\mathbb{T}$. The *global passage time* random variable is the time T at which the population of agents in the target set reaches p for the first time, that is $T \stackrel{def}{=} \inf\{t \in \mathbb{R}_+ \mid \sum_{i \in \mathbb{T}} X_i \geq p\}$.

In Section 3.6 we show that under an increasing system scale, the population process of a PCTMC converges to the deterministic mean-field approximation. Therefore, the following point-mass approximation can be used to approximate the global passage time:

$$T \approx \inf\left\{t \in \mathbb{R}_+ \mid \sum_{i \in \mathbb{T}} \widetilde{\mathbb{E}}[X_i] \geq p\right\}$$

Another possibility is to use the moments of populations to approximate the CDF of T. For example, the Markov inequality expresses bounds on the CDF as functions of population means:

$$\mathbb{P}(T \leq t) \leq \frac{n - \sum_{i \in \mathbb{T}} \widetilde{\mathbb{E}}[X_i(t)]}{n - p}$$
$$\mathbb{P}(T \leq t) \geq 1 - \frac{\sum_{i \in \mathbb{T}} \widetilde{\mathbb{E}}[X_i(t)]}{p + 1}$$

A tighter approximation can be obtained by using the Chebyshev's inequality and second-order moments of populations:

$$\mathbb{P}(T \leq t) \leq \frac{\widetilde{\mathrm{Var}}[\sum_{i \in \mathbb{T}} X_i(t)]}{\widetilde{\mathrm{Var}}[\sum_{i \in \mathbb{T}} X_i(t)] + (\widetilde{\mathbb{E}}[\sum_{i \in \mathbb{T}} X_i(t)] - p)^2}$$
$$\mathbb{P}(T \leq t) \geq 1 - \frac{\widetilde{\mathrm{Var}}[\sum_{i \in \mathbb{T}} X_i(t)]}{\widetilde{\mathrm{Var}}[\sum_{i \in \mathbb{T}} X_i(t)] + (\widetilde{\mathbb{E}}[\sum_{i \in \mathbb{T}} X_i(t)] - p - 1)^2}$$

3.6 Convergence

Although Theorem 1 gives an exact form of moment ODEs, the mean-field and higher-order moment ODEs, Section 3.4.1 and Section 3.4.2, are obtained after applying various approximations. The resulting moments can be numerically accurate, but there is little understanding of the total error. One way to more formally justify the ODE analysis techniques is to show an agreement between the exact moments and the approximation in a limit of increasing initial populations. In case of the mean-field analysis, we present a known result that shows that population means converge to the ODE approximation as initial populations increase (and the transition rates are scaled appropriately). For higher-order approximations, an existing result shows that in a similar limit of increasing initial populations, a PCTMC can be expressed as a linear combination of the mean-field solution and a Gaussian process. This results in a stochastic differential equation, with a set of ODEs capturing the evolution of the covariance of the Gaussian process. These ODEs turn out to agree with Theorem 1, closed under a certain approximation, providing a heuristic justification for the convergence of second-order moment ODE approximations from Section 3.4.2. Moreover, the Gaussian representation justifies the use of the normal approximation given by Theorem 2.

In order to make the desired theoretical considerations in this section, it is necessary to consider a *density dependent* sequence of PCTMCs which have the same population structure and transition rate difference vectors and differ only in their initial populations and rates of transition classes, as defined below. We start with a PCTMC $(\boldsymbol{X}(t), \mathcal{C}, \boldsymbol{x_0})$ and define a sequence of PCTMCs indexed by a scale factor $S \in \mathbb{Z}_+$: for each value of S, we define $(\boldsymbol{X}^{(S)}(t), \mathcal{C}^{(S)}, \boldsymbol{x}_\mathbf{0}^{(S)})$ where

(i) the population structure of $\boldsymbol{X}^{(S)}(t)$ is the same as that of $\boldsymbol{X}(t)$,

(ii) for each transition class $c \in \mathcal{C}$, there is a corresponding transition class $c^{(S)} \in \mathcal{C}^{(S)}$ such that $r_{c^{(S)}}(\boldsymbol{x}) = S \cdot r_c(\boldsymbol{x}/S)$ for any population vector $\boldsymbol{x}$,

(iii) the initial populations are scaled by S, that is $\boldsymbol{x}_\mathbf{0}^{(S)} = S \cdot \boldsymbol{x_0}$.

For example, if the PCTMC $\boldsymbol{X}(t)$ corresponds to a GPEPA model, the condition (ii) holds due to the homogeneity property from Equation 3.9.

For each $\boldsymbol{X}^{(S)}(t)$ in the sequence, we can obtain the mean-field approximation, Equation 3.6, approximating the evolution of $\mathbb{E}[\boldsymbol{X}^{(S)}(t)]$ with a real-valued vector $\boldsymbol{x}^{(S)}(t)$, described by a system of ODEs, Equation 3.6:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}^{(S)}(t) &= \boldsymbol{f}^{(S)}(\boldsymbol{x}^{(S)}(t)) \\ \boldsymbol{x}^{(S)}(0) &= \boldsymbol{x}_\mathbf{0}^{(S)} \end{aligned}$$

Because of the density dependent properties, solutions to these systems of ODEs only differ by the scale factor S. This allows us to define a *rescaled mean-field* approximation, as $\bar{x}(t) = \boldsymbol{x}^{(S)}(t)/S$, independently of S. We also define a rescaled version of the stochastic process, dependent on S: $\bar{\boldsymbol{X}}^{(S)}(t) = \boldsymbol{X}^{(S)}(t)/S$.

3.6.1 Convergence of mean approximations

Hayden et al. [4] generalise a convergence result originally due to Darling and Norris [63] to non-smooth rates in GPEPA models:

Theorem 3. In the notation introduced above, for fixed $t_f \geq 0$ and $\varepsilon > 0$, as $S \to \infty$:

$$P\left(\sup_{t\in[0,t_f]} \left\|\bar{X}^{(S)}(t) - \bar{x}(t)\right\| > \varepsilon\right) \to 0$$

uniformly for $t \in [0, t_f]$ for all deterministic initial states $\boldsymbol{x_0}$. Moreover, the convergence holds if PCTMCs in the sequence have random initial populations $\boldsymbol{X}_{\boldsymbol{0}}^{(S)}$ if for all $\delta > 0$, $P(\|\boldsymbol{X}_{\boldsymbol{0}}^{(S)}/S - \boldsymbol{x}_{\boldsymbol{0}}^{(S)}/S\| > \delta) \to 0$ as $S \to \infty$.

3.6.2 Convergence of variance approximation

Similar to the mean convergence above, Hayden and Bradley [100] generalise a result originally due to Kurtz [128] to GPEPA models and therefore to PCTMCs with minimum rates. This result suggests the following approximation to a PCTMC:

$$\boldsymbol{X}^{(S)}(t) \approx S\bar{\boldsymbol{x}}(t) + \sqrt{S}\boldsymbol{E}(t) \tag{3.19}$$

where $\boldsymbol{E}(t)$ is a continuous state-space Gaussian process. The stochastic process $\boldsymbol{E}(t)$ is given in the following theorem [100], which applies to a sequence of density dependent PCTMC $\{\boldsymbol{X}_S(t)\}_{S=1}^{\infty}$ and rescaled mean-field approximation $\bar{\boldsymbol{x}}(t)$. Let $K = |\mathcal{C}|$.

Theorem 4. Let $t_f > 0$ and let $\hat{\boldsymbol{T}}$ be the subset of $\{t \in [0, t_f)\}$ for which $\boldsymbol{f}(\cdot)$ is not totally differentiable at the point $\bar{\boldsymbol{x}}(t)$. We require that $\hat{\boldsymbol{T}}$ has Lebesgue measure zero. Then on all of $[0, T) \setminus \hat{\boldsymbol{T}}$, $\boldsymbol{f}(\cdot)$ has a well-defined Jacobian at the point $\bar{\boldsymbol{x}}(t)$, say $D\boldsymbol{f}(\bar{\boldsymbol{x}}(t))$. Extend this to all points $\{\bar{\boldsymbol{x}}(t) : t \in [0, t_f)\}$, say by defining it to be the matrix of zeros at times in $\hat{\boldsymbol{T}}$.

Then as $S \to \infty$, $\frac{\boldsymbol{X}^{(S)}(t)}{\sqrt{S}} - \sqrt{S}\bar{\boldsymbol{x}}(t) \Rightarrow \boldsymbol{E}(t)$, where:

$$\boldsymbol{E}(t) := \int_0^t D\boldsymbol{f}(\bar{\boldsymbol{x}}(s)) \cdot \boldsymbol{E}(s)\mathrm{d}s + \sum_{c\in\mathcal{C}} B_c\left(\int_0^t r_c(\bar{\boldsymbol{x}}(s))\mathrm{d}s\right)\boldsymbol{\delta}_c$$

and $\{B_k(t)\}_{k=1}^{K}$ are K mutually independent Brownian motions. The convergence is weak convergence on $D_{\mathbb{R}_+^N}[0, t_f)$, the space of $\mathbb{R}_+^N$-valued, right continuous with left limits, functions, equipped with the uniform topology.

The authors note that the process $\boldsymbol{E}(t)$ is the unique solution of the following (Itō) stochastic differential equation (SDE):

$$\mathrm{d}\boldsymbol{E}(t) = \boldsymbol{\mu}(\boldsymbol{E}(t), t)\mathrm{d}t + \boldsymbol{\sigma}(t)\mathrm{d}\boldsymbol{B}(t)$$

where $\boldsymbol{\mu} : \mathbb{R}^N \times \mathbb{R}_+ \to \mathbb{R}^N$ and $\boldsymbol{\sigma} : \mathbb{R}_+ \to \mathbb{R}^{N\times K}$ are defined by:

$$\boldsymbol{\mu}(\boldsymbol{y}, t) \stackrel{def}{=} D\boldsymbol{f}(\bar{\boldsymbol{x}}(t)) \cdot \boldsymbol{y}$$

$$\boldsymbol{\sigma}(t) \stackrel{def}{=} \left(\boldsymbol{\delta}_{j,i} \times \sqrt{r_j(\bar{\boldsymbol{x}}(t))}\right)_{ij}$$

and $\boldsymbol{B}(t) \stackrel{def}{=} (B_1(t), \ldots, B_K(t))^{\mathrm{T}}$ is a K-dimensional standard Brownian motion. This representation allows them to apply the machinery of Itō's Lemma [e.g. 116, Theorem 17.18] to derive the following system of ODEs, whose unique solution is exactly the covariance matrix of $\boldsymbol{E}(t)$:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{Cov}[\boldsymbol{E}(t), \boldsymbol{E}(t)] = \mathrm{Cov}[\boldsymbol{E}(t), \boldsymbol{E}(t)](D\boldsymbol{f}(\bar{\boldsymbol{x}}(t)))^{\mathrm{T}} + D\boldsymbol{f}(\bar{\boldsymbol{x}}(t))\mathrm{Cov}[\boldsymbol{E}(t), \boldsymbol{E}(t)]^{\mathrm{T}} \\ + \sum_{c \in \mathcal{C}} r_c(\bar{\boldsymbol{x}}(t))\boldsymbol{\delta}_c(\boldsymbol{\delta}_c)^{\mathrm{T}}\end{aligned} \tag{3.20}$$

If we apply this to the client–server example for the specific variance of the *Server* population, $E_S(t)$, we have:

$$\begin{aligned}&\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{Cov}[E_S(t), E_S(t)] = \\ &\quad - 2r_{request}\left(\mathbf{1}_{\{\bar{x}_S(t) \leq \bar{x}_C(t)\}}\mathrm{Cov}[E_S(t), E_S(t)] + \mathbf{1}_{\{\bar{x}_S(t) > \bar{x}_C(t)\}}\mathrm{Cov}[E_S(t), E_C(t)]\right) \\ &\quad + 2r_{data}\left(\mathbf{1}_{\{\bar{x}_{S_g}(t) \leq \bar{x}_{C_w}(t)\}}\mathrm{Cov}[E_S(t), E_{S_g}(t)] + \mathbf{1}_{\{\bar{x}_{S_g}(t) > \bar{x}_{C_w}(t)\}}\mathrm{Cov}[E_S(t), E_{C_w}(t)]\right) \\ &\quad - 2r_{break}\mathrm{Cov}[E_S(t), E_S(t)] + 2r_{reset}\mathrm{Cov}[E_S(t), E_{S_b}(t)] \\ &\quad + r_{request}\min(\bar{x}_C(t), \bar{x}_S(t)) + r_{data}\min(\bar{x}_{C_w}(t), \bar{x}_{S_g}(t)) + r_{break}\bar{x}_S(t) + r_{reset}\bar{x}_{S_b}(t)\end{aligned} \tag{3.21}$$

Note that Theorem 4 suggests the approximation:

$$\mathrm{Cov}[S(t), S(t)] \approx \mathrm{Cov}[S\bar{x}_S(t) + \sqrt{S}E_S(t), S\bar{x}_S(t) + \sqrt{S}E_S(t)] = S\mathrm{Cov}[E_S(t), E_S(t)]$$

In general, Theorem 4 implies that (assuming its hypothesis), as $S \to \infty$,

$$\frac{1}{S}\mathrm{Cov}[\boldsymbol{X}^{(S)}(t), \boldsymbol{X}^{(S)}(t)] \to \mathrm{Cov}[\boldsymbol{E}(t), \boldsymbol{E}(t)].$$

The system of ODEs given in Equation 3.20 yields an approximation to the covariance matrix of the underlying PCTMC of a general GPEPA model. Furthermore, this approximation is guaranteed by Theorem 4 to converge in the limit of large populations.

However, the covariance ODE approximation from Equation 3.11 consists of integrating a slightly different system of ODEs. These are very similar to those of Equation 3.20 and in fact can intuitively be regarded as a better approximation to the actual covariance matrix. This is the basis of our conjecture that a similar convergence result also holds, and furthermore, that the rate of convergence may well be faster for the ODEs from Equation 3.11.

Theorem 1 gives the following differential equation for $\mathrm{Cov}[S(t), S(t)]$:

$$\begin{aligned}&\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{Cov}[S(t), S(t)] = \frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[S^2(t)] - 2\mathbb{E}[S(t)]\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[S(t)] = \\ &\quad - 2r_{request}\left(\mathbb{E}[\min(C(t)S(t), S^2(t))] - \mathbb{E}[\min(C(t), S(t))]\mathbb{E}[S(t)]\right) \\ &\quad + 2r_{data}\left(\mathbb{E}[\min(C_w(t)S(t), S_gS(t))] - \mathbb{E}[\min(C_w(t), S_g(t))]\mathbb{E}[S(t)]\right)\end{aligned}$$

$$- 2r_{break}\left(\mathbb{E}[S^2(t)] - \mathbb{E}^2[S(t)]\right) + 2r_{reset}\left(\mathbb{E}[S_b(t)S(t)] - \mathbb{E}[S_b(t)]\mathbb{E}[S(t)]\right)$$
$$+ r_{request}\mathbb{E}[\min(C(t), S(t))] + r_{data}\mathbb{E}[\min(C_w(t), S_g(t))] + r_{break}\mathbb{E}[S(t)] + r_{reset}\mathbb{E}[S_b(t)] \tag{3.22}$$

Since the corresponding system of ODEs cannot be solved analytically or numerically due to the presence of expectations of non-linear functions, the first-order approximations as given by Equation 3.5 and second-order approximations given by Equation 3.10 can be applied repeatedly as in Section 3.4.2 to deduce approximate ODEs for $\mathrm{Cov}[S(t), S(t)]$ and the other covariances. This system of ODEs can then be solved numerically.

The key point to note now is that if we keep the first-order approximations the same but replace the second-order approximations by ones of the form:

$$\mathbb{E}[\min(C(t)S(t), S^2(t))] \approx \mathbf{1}_{\{\mathbb{E}[S(t)]\leq\mathbb{E}[C(t)]\}}\mathbb{E}[S^2(t)] + \mathbf{1}_{\{\mathbb{E}[S(t)]>\mathbb{E}[C(t)]\}}\mathbb{E}[C(t)S(t))]$$

then we recover the system of ODEs of Equation 3.20. This is a reasonable approximation, but we evaluate a minimum of second-order terms by making only first-order comparisons. Intuitively, this is likely to be a worse approximation than the original ODE analysis approach. Therefore we can heuristically justify that the covariance ODE approximations should converge to the actual covariances when scaled by the scale factor S.

3.7 Conclusion

The main contribution of this chapter is the definition of Population Continuous-Time Markov Chains (PCTMC), a unified representation for Markov population models. This representation is separated from the behavioural description of the model. For example, we defined the semantics of the GPEPA process algebra in terms of an underlying PCTMC. An advantage of this approach is that all analysis techniques for PCTMCs are readily available to behavioural models that can be translated to a PCTMC. We have shown how the rapid ODE analysis applies to PCTMCs and also how to use the ODE analysis to obtain approximation of passage times in PCTMCs. We are now equipped with a framework that can rapidly analyse large-scale population models, described in a behavioural language, and provide access to passage-time metrics which are crucial in verification of SLAs. In the rest of this thesis, we develop this framework further. We show how to extend the analysis to compute metrics related to energy consumption and environment temperature and thus accurately address the performance–energy trade-off outlined in Chapter 1.

Chapter 4

Improving accuracy of ODE analysis of PCTMCs

Key contributions		
Investigation of accuracy of ODE analysis	4.2	[8]
Hybrid simulation of PCTMC models	4.3	[13]
Normal moment closure for minimum rates	4.4	[3, 6]
Evaluation of accuracy of moment closures	4.5	[3]

4.1 Introduction

In this chapter, we investigate the accuracy of the ODE analysis of PCTMCs as described in Section 3.4. Theorem 3 guarantees that the solution to the system of ODEs approximating means of populations in a PCTMC $\boldsymbol{X}(t)$ converges to the exact population means of the stochastic process as the system scale increases – the quantities $\widetilde{\mathbb{E}}[\boldsymbol{X}^{(S)}(t)]$ converge to $\mathbb{E}[\boldsymbol{X}^{(S)}]$ as S increases. However, in practice, modellers have to work with a range of finite system scales for which they need to obtain accurate quantitative predictions. In such cases, the above convergence results do not provide any guarantees about the accuracy of the ODE approximation. Several techniques, such as that of [63], provide *bounds* on the distance between the stochastic process and deterministic approximation. These bounds are often loose and not applicable in practice. A recent result by Bortolussi and Hayden [38] provides steady-state and transient bounds on the dynamics of a discrete-time Markov Chain and demonstrates that these can be much tighter than the best previously known bounds.

In this chapter, we take a heuristic approach and introduce a method that can reliably predict whether the mean and higher-order moment ODEs give a good approximation to moments in a PCTMC with rates containing the minimum function. Based on this heuristic, we are able to design a hybrid approach combining the ODE analysis and stochastic simulation. This, in turn, leads to a new moment closure approximation that greatly improves the accuracy of moment ODEs. We conclude the chapter by evaluating the approximation error over a large number of parameters for several models.

Ultimately, the main aim of this chapter is to serve as a solid foundation for further developments in this thesis. In later chapters, we will extend the ODE analysis with equations capturing additional metrics and therefore it is crucial to have a good understanding of the nature of the approximations of the core ODE system. In particular, in Chapter 6 we will introduce a new class

of reward-controlling PCTMCs, resulting in a behaviour that keeps the system in a mode that is exactly where the heuristic method predicts a large error. We will demonstrate that only the improved moment closure approximation makes it possible to efficiently analyse higher-order moments in such models.

The main contributions of this chapter can be summarised as the following:

Switch-point analysis of models with minimum functions Initially, we focus on PCTMC models with rates containing occurrences of the minimum function, such as those described by the GPEPA stochastic process algebra or queueing network models. Extending the original work of Hayden [98], in Section 4.2 we propose a method that uses the so-called *switch-point distance* in a PCTMC to identify situations where there is a likely error in the ODE approximation. A similar approach has been used by Zhao [192], where the author evaluates a closed form expression to estimate a parameter region where an ODE approximation to a closed queueing networks achieves a high error.

Hybrid simulation of PCTMCs In Section 4.3 we show a first possible improvement in the accuracy of ODE analysis by combining simulation and the ODE approximation. We use the switch-point distance from the previous section to decide when to apply ODE analysis and simulation respectively.

Normal moment closure for minimum rates In Section 4.4 we adapt the moment closure techniques from Section 3.4.3 to work on rate functions containing the minimum function. This results in a system of ODEs where, unlike in previous approaches [98], the equations for means use the variance approximations, in particular to define the expected behaviour around switch-points. This closure greatly improves the accuracy of the ODE analysis and can be reliably used in analysing a large number of models.

Validation of accuracy of moment closures In Section 4.5 we numerically evaluate the accuracy of moment ODEs on a large number of different models. We compare the approximations to confidence interval estimates obtained from exact stochastic simulation.

4.2 Switch-point analysis in PCTMC models with minimum rates

The nature of the approximation of ODEs for moments of populations in a PCTMC $(\boldsymbol{X}(t), \mathcal{C}, \boldsymbol{X_0})$ depends on the rate functions of transition classes in $\mathcal{C}$. For example, the right-hand side of the exact mean equations, Equation 3.4, consists of terms $\mathbb{E}[r_c(\boldsymbol{X}(t))]$ for each transition class c. If the rate function $r_c(\cdot)$ is a constant function or a linear combination of populations, we say that the PCTMC is *purely concurrent*. In such cases the expectation can be expanded in terms of population means, which have a corresponding differential equation in the system and the resulting system of ODEs can be solved analytically and provide the *exact* means.

However, in general, it is not possible to exactly expand the expectation as a linear combination of population means (or higher-order moments) and therefore an approximation has to be introduced. In the case of GPEPA, the PCTMC semantics from Section 3.2.2 determines the form

of the transition rates. If all cooperation action sets in a model are empty, we get the purely concurrent case above. Otherwise the transition rates are of the form

$$r(\boldsymbol{X}(t)) = f(\boldsymbol{X}(t)) \cdot \min(r_1(\boldsymbol{X}(t)), r_2(\boldsymbol{X}(t))) \tag{4.1}$$

where the functions r_1 and r_2 are apparent rates in smaller sub-models and f is a rational function of transition rates and apparent rates (Section 3.2.2). In many cases the rates are less complex – for example the *split-free* class of GPEPA models, introduced by Hayden and Bradley [99], allows only one possible cooperation for each action label. For example, the client–server model from Section 3.2.3 is an instance of a split-free model. On the other hand the probed version of the same model in Section 3.5.2 is not, due to the multiple versions of the client agent (plain and with an ongoing and finished probe agent) both offering the $request$ and $data$ actions.

The assumption of split-free models simplifies the above form so that the function f is constant. The mean equations depend on the approximation from Equation 3.5, which in case of minimum functions gives

$$\mathbb{E}[\min(r_1(\boldsymbol{X}(t)), r_2(\boldsymbol{X}(t)))] \approx \min(r_1(\mathbb{E}[\boldsymbol{X}(t)]), r_2(\mathbb{E}[\boldsymbol{X}(t)])).$$

Note that the functions $r_1(\cdot)$ and $r_2(\cdot)$ may also include further minimum terms themselves and thus induce multiple further applications of the approximation not shown explicitly above. Hayden and Bradley [99] argue that the error of this approximation is at its highest around the points where the arguments to the minimum functions are equal, naming each such point a *switch point*. The switch points occur when the total rate of cooperation between agent groups becomes equal, that is $r_1(\boldsymbol{X}(t)) = r_2(\boldsymbol{X}(t))$, causing the minimum function to switch between its arguments.

One of the main contributions of this section is to verify and further investigate this claim empirically. We show how to use the solution to the moment ODEs to evaluate the *distance* of the model from a switch point. We define the *switch-point distance* as the difference between the two arguments of a minimum term in a rate function. We demonstrate that at times when this distance is likely to be near zero, an error is introduced into the approximation. We show how the variance of the switch-point distance obtained from moment ODEs can be used to detect when the distance is near zero. This will be the first instance of a common theme in this chapter, where we show different ways of using the second-order moments of populations to obtain a more accurate approximation of the first order moments of populations.

4.2.1 Numerical investigation: GPEPA client–server model

In this section, we empirically investigate the nature of ODE moment approximations from Section 3.4 and the nature of convergence results from Section 3.6. We will demonstrate, on an example model, that simulation means converge to the ODE approximations as the system scale increases, as defined in Section 3.6. We also look at higher moments to observe the second-order convergence result from Section 3.6.2. We quantitatively examine the error of ODE approximation at different scales of the system and evaluate the error in the context of switch-point distance.

Consider the client–server model from Section 3.2.3 under two sets of parameter regimes – one that results in the model not staying near a switch point in any long time interval (we will informally call this the *occasionally switching* model, model A) and one that forces the model to steadily stay near a switch point for a longer period of time (we will informally call this the *persistently switching* model, model B).

In both cases we fix the initial client population at $n_C = 100$ and the server population at $n_S = 50$. The two sets of rate parameters are shown in Table 4.1

Table 4.1: Two sets of rate parameters for the client–server model.

	$r_{request}$	r_{break}	r_{think}	r_{data}	r_{reset}
Model A	2.0	0.1	0.20	1.0	2.0
Model B	2.0	0.3	0.35	2.0	0.05

There are two possible sources of switch points in the client–server model, each corresponding to an instance of cooperation in the model. One, the term $\mathbb{E}[\min(C(t), S(t))]$, comes from the cooperation when a client establishes a connection with a server. Another, $\mathbb{E}[\min(C_w(t), S_g(t))]$, comes from the cooperation when a client retrieves data from an available server.

For the minimum term involving populations of the *Server_get* and *Client_wait* states, the distance between its two arguments stays at zero all the time. The two corresponding population processes are stochastically identical and so the minimum function can be reduced to either of its arguments.

Figure 4.1 shows the switch-point distance in these two models resulting from the minimum term involving populations of *Client* and *Server* agents. As the time progresses, the offered service rate by servers becomes equal to the demand rate of clients. When the means of the two rates are equal, the system reaches a switch point. From Figure 4.1a we can see that Model A hits one switch point at time $t \simeq 2.1$. Figure 4.1 also shows an interval of width 2.58 standard deviations around the mean switch-point distance, representing the interval where the distance lies with probability 0.99 under the assumption that the distribution of the switch-point distance is approximately normal. For model A, the distance is unlikely to be near zero apart from the time interval $\approx (1.7, 3.0)$. As seen in Figure 4.1b, Model B hits two switch points when $t \simeq 2.8$ and $t \simeq 4.8$ and stays within the 99% interval for a longer time interval $\approx (1.7, 12.0)$.

We investigate the error in the moment approximation for the above models and compare them in the context of the shown switch-point distance plots. We look at expectation, variance and skewness (the standardised third-order moment) of all populations for each model and its respective versions with initial populations scaled by a factor of $S = 1, 4, 16$ and 64. We investigate how the scale influences the error. For each scale, we plot the difference between moments approximated by ODEs and their exact estimates from stochastic simulation. We relate the error to the switch-point distance plots and suggest that a small switch-point distance can predict an increase in the error in means and higher-order moments. In many cases, it is sufficient to use an approximate switch-point distance from the ODE analysis itself to evaluate its accuracy.

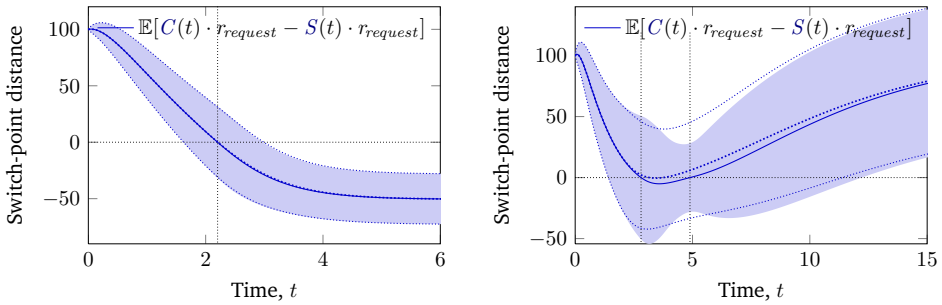
Model A: Occasionally switching model

For the combination of rate parameters in Model A, the client–server model does not stay near a switch point for a longer period of time. We start by comparing side by side, for mean, variance and skewness, the simulation estimate with the respective ODE approximation in Figure 4.2.

For all three moments, the ODE approximation is qualitatively close to the simulation estimates. However, there are visible quantitative differences for variance and skewness. These are especially noticeable for moments involving the population of *Client* agents, in the time interval around the switch point from Figure 4.1.

According to Theorem 3, the error in mean populations shown in Figure 4.2 becomes negligible as the scale of the PCTMC increases. Theorem 4 suggests a similar convergence for second-order moments. We empirically verify these claims by plotting the errors in mean and standard deviation in the population of client agents in a scaled version of the model with $S = 1, 4, 16$ and 64. The error is calculated as the difference between the moments from simulation and ODE analysis. Additionally, we normalise the error in means and standard deviations in accordance with Theorem 3 i.e. divide the error by S. Figure 4.3d shows the switch-point distance plot at different scales of the system.

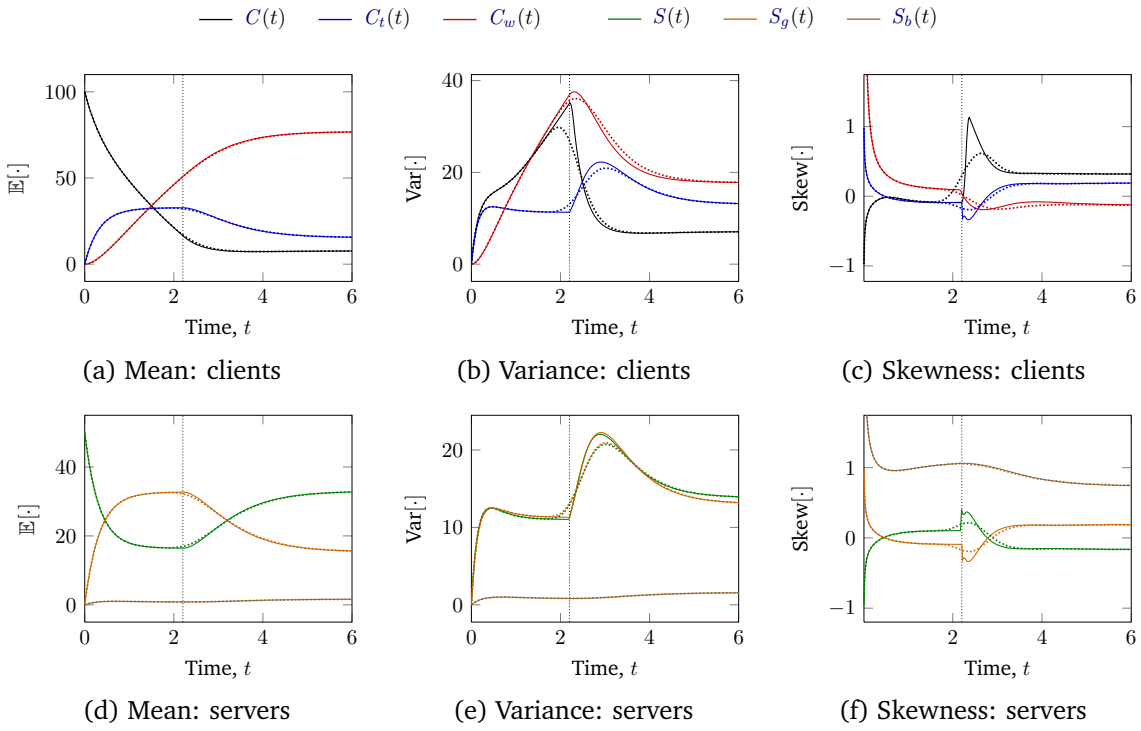
Figure 4.3 agrees with the statements of Theorem 3 and Theorem 4. Moreover, as shown in the skewness plot in Figure 4.3c, a similar property might hold for third-order moments. Neither of the convergence properties quantifies the error at different points in time. However, as discussed in the previous section, the switch-point distance plot can give an indication of the accuracy of the approximations. At all scales, the error is nearly zero when the switch-point distance is not likely to be zero – the approximate 99% interval does not contain the zero value. As the system approaches a switch point, errors start increasing. These times can be seen on the switch-point distance plot, Figure 4.3d, as points where the shaded regions cross the time axis. The error decreases and returns to near zero some time after the switch point distance increases.



(a) Switch-point distance for Model A

(b) Switch-point distance for Model B

Figure 4.1: Switch-point distance plot for the client–server model. The shaded region shows an approximate 99% interval. The dotted lines show the respective mean and standard deviation intervals obtained from 10^7 replications of simulation of the models.



(a) Mean: clients (b) Variance: clients (c) Skewness: clients

(d) Mean: servers (e) Variance: servers (f) Skewness: servers

Figure 4.2: Moments of populations in the client–server model, Model A. The solid lines are approximations by the ODE analysis and dotted lines estimates from 10^7 replications of stochastic simulation. The vertical lines correspond to the switch point from Figure 4.1a.

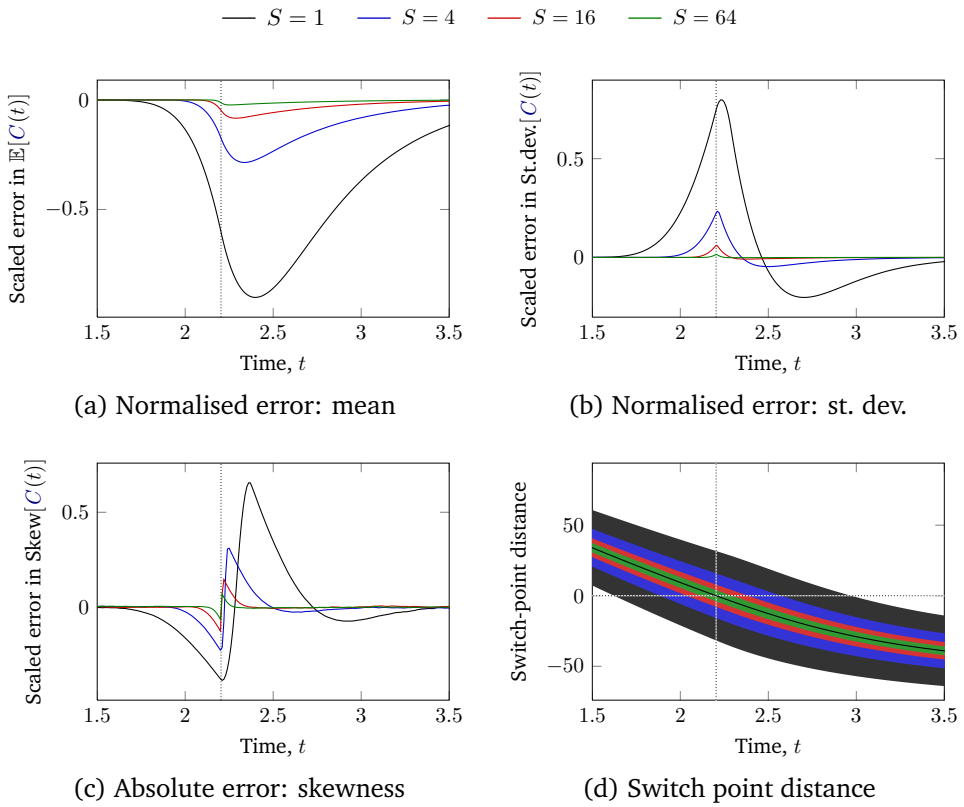
Model B: Persistently switching model

In case of the parameter values of Model B, the client–server model goes through a much longer time interval with a small switch-point distance, Figure 4.1b. Figure 4.4 plots the mean and variance of client and server populations. Both mean and variance approximations are visibly larger than in case of Model A. The error concentrates in an interval where the switch-point distance is relatively small.

Figure 4.5 looks at the error more closely and plots the difference between moments from simulation and their ODE approximation for *Client* population, for different scales of Model B. It can be seen that the normalised error in Figure 4.5 is higher than in the case of Model A. This is caused by the fact that the switch-point distance stays near zero for a longer period of time. We can confirm that the error for mean and variance seems to be going to zero in the scale limit. Same as for Model A, the time when the error starts increasing for the first time can be predicted from the switch-point distance plot, Figure 4.5d.

4.2.2 Discussion

The presented examples give a guideline for how to predict errors in the efficient ODE analysis of PCTMC models with minimum rates without running computationally more expensive stochastic simulations. We observed, for a model where the resulting differential equations are piecewise linear, that the error is influenced by the switch-point distance. That is, how closely the model stays near a switch point – the situation where the arguments of the minimum rate functions are relatively close. If the model only crosses switch points at certain points of time and does



(a) Normalised error: mean

(b) Normalised error: st. dev.

(c) Absolute error: skewness

(d) Switch point distance

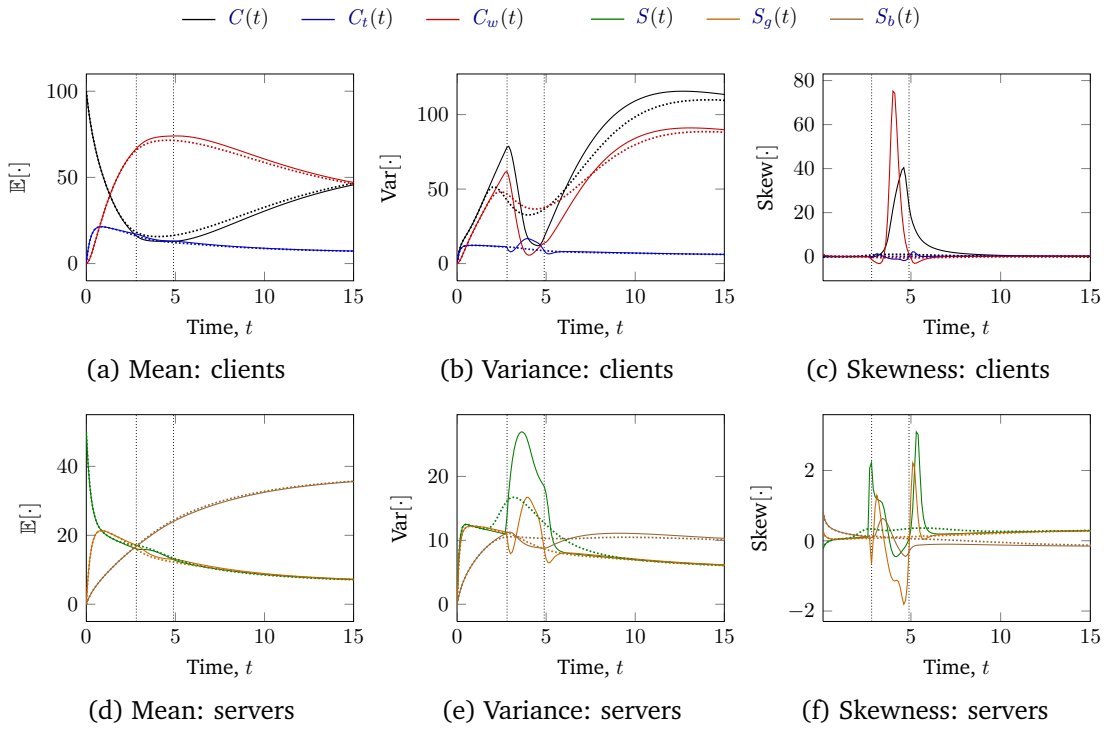
Figure 4.3: Effects of scaling on normalised error and switch point distance in Model A. Figure (d) shows the scaled mean switch point distance as the thick black line, which is invariant with the scale. The respective approximate 99% intervals are displayed as shaded regions around the mean. These are drawn over each other as the width of the interval decreases with increasing scale.

not stay near any during the rest of the time, then the error is concentrated tightly around those times. The general strategy is the following:

1. Solve the underlying system of ODEs for first- and second-order moments, $\widetilde{\mathbb{E}}[X_i(t)]$ and $\widetilde{\mathbb{E}}[X_j(t)X_i(t)]$ respectively for $1 \leq i, j \leq N$.
2. Identify all rates of the form $\min(r_1(\boldsymbol{X}(t)), r_2(\boldsymbol{X}(t)))$.
3. Use the first-order moments of populations to compute the mean of switch-point distance $\widetilde{\mathbb{E}}[d(\boldsymbol{X}(t))]$ where $d(\boldsymbol{X}(t)) = r_1(\boldsymbol{X}(t)) - r_2(\boldsymbol{X}(t))$ for t within the desired time interval. Use the second-order moments to compute $\widetilde{\mathrm{Var}}[d(\boldsymbol{X}(t))]$.
4. The first time when the interval bounded by $\widetilde{\mathbb{E}}[d(\boldsymbol{X}(t))] \pm 2.58 \cdot \widetilde{\mathrm{Var}}[d(\boldsymbol{X}(t))]^{1/2}$ contains zero is when an error of the approximation is likely. When the interval stays away from zero for a longer period of time, it is possible that the error decreases back to zero.

This also takes into account the scale of the system. As the scale increases, standard deviation of the switch-point distance gets relatively smaller and the 99% interval stays away from zero for longer periods of time.

In practice, the method could be used to more reliably apply the ODE analysis in large-scale parameter explorations. For example, in the client–server model, we can be interested in the optimal number of servers that guarantees some derived performance metrics. For each parameter



(a) Mean: clients (b) Variance: clients (c) Skewness: clients

(d) Mean: servers (e) Variance: servers (f) Skewness: servers

Figure 4.4: Moments of populations in the client–server model, Model B. The solid lines are approximations by the ODE analysis and dotted lines estimates from 10^6 replications of stochastic simulation. The vertical lines correspond to the switch points from Figure 4.1b.

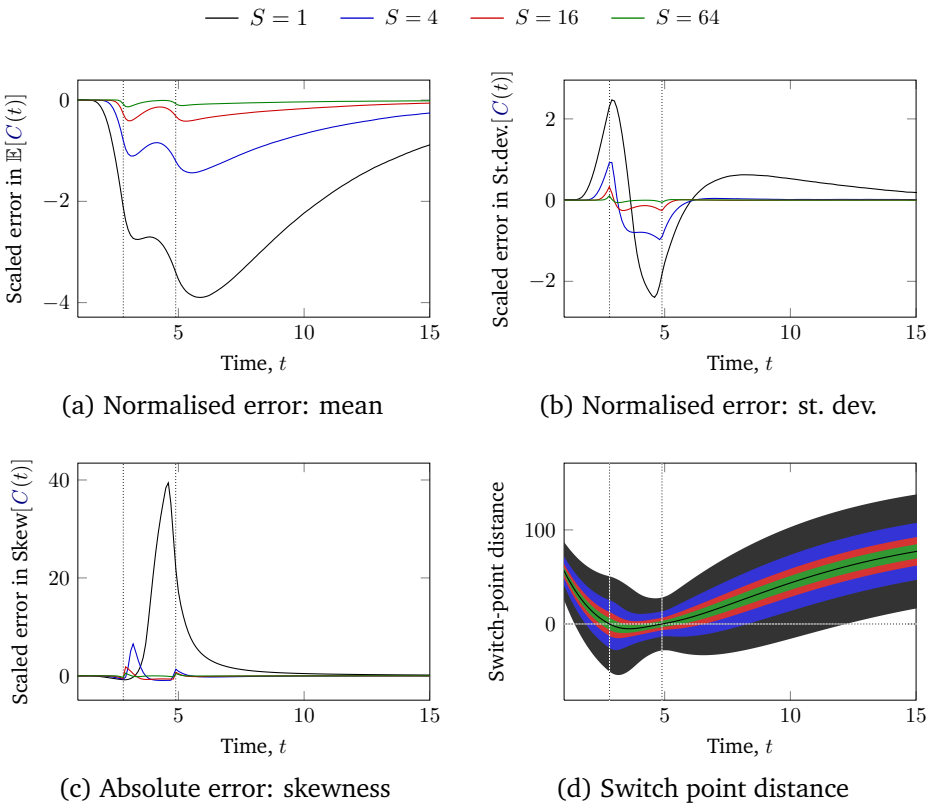
combination, we can first cheaply analyse the PCTMC with ODEs. We can use the switch-point distance to predict if an error is likely. In that case, we can analyse the system exactly with stochastic simulation. Alternatively, we can use the ODE analysis to find an optimal configuration and then verify the derived metrics with stochastic simulation.

The main limitation is that the switch-point distance method does not *improve* the accuracy of the ODE analysis. In most situations, a low switch-point distance is unavoidable. In fact, large switch-point distance is often a sign of under-utilisation of resources and parameter configurations achieving small switch-point distance are sought. In the next section, we provide a simple improvement which combines the ODE analysis with simulation.

4.3 First improvement: combining ODE analysis and simulation

In the previous section, we proposed a way to detect where the ODE analysis fails to accurately capture the evolution of moments in a PCTMC model. In such cases, it would be necessary to run stochastic simulations to properly understand the system behaviour. In this and the following section, we try to improve this situation by reducing, or completely eliminating, the need for simulation.

As a first improvement, we combine the ODE analysis with stochastic simulation in a way that gives better control over the accuracy–computational cost trade-off. The switch-point distance heuristic can be used to suggest time intervals during which the ODE analysis is likely to be less



(a) Normalised error: mean

(b) Normalised error: st. dev.

(c) Absolute error: skewness

(d) Switch point distance

Figure 4.5: Influence of scaling on the normalised error and the switch point distance in Model B. Figure (d) shows the scaled mean switch point distance as the thick black line, which is invariant with the scale. The respective approximate 99% intervals are displayed as shaded regions around the mean. These are drawn over each other as the width of the interval decreases with increasing scale.

accurate. During these time intervals, stochastic simulation replaces ODEs, giving rise to a type of *hybrid analysis*. Figure 4.6 shows an overview of the analysis.

Each such hybrid analysis must address several issues. It has to choose which ODEs are included in the system. The analysis also must facilitate the transfer from ODEs to simulations at each desired time t_s and the transfer from simulation to ODEs at each time t_o. We describe two ways to tackle this problem.

The efficiency of the hybrid analysis depends on the length of the simulation interval. Usually, the cost of numerically solving the ODEs is negligible compared to running sufficiently many replications of the simulation. The length of the simulation intervals depends on the switch

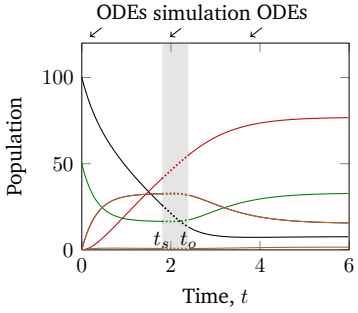
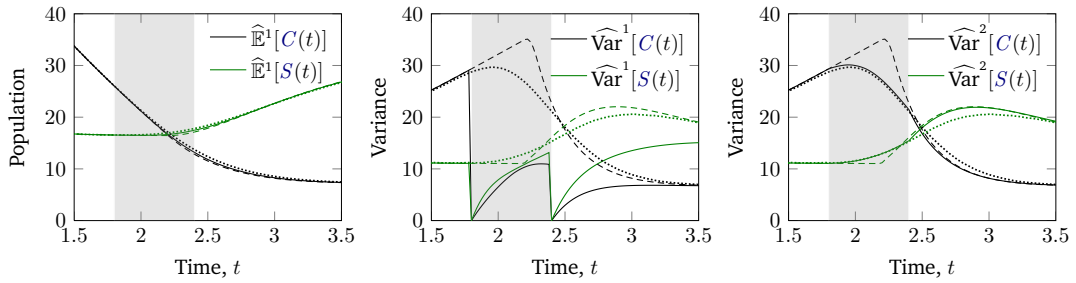


Figure 4.6: Overview of the hybrid analyses.

point behaviour. For many of the examples below, this interval is only one tenth of the total time considered, thus giving a near ten-fold improvement over analysis driven entirely by stochastic simulation.

4.3.1 First-order hybrid analysis

The *first-order hybrid analysis* combines the ODEs for mean populations with stochastic simulation to produce more accurate approximations of the mean populations denoted by $\widehat{\mathbb{E}}^1[\cdot]$. At each time t_s where simulation replaces the ODEs, each replication of the simulation is restarted with populations deterministically set to the values given by the solution to the ODEs at time t_s. Note that these can be real-valued and so the simulated stochastic process is extended accordingly. At each time t_o where the analysis returns back to the ODEs after simulation, the initial values of the mean ODEs are set to the means from the simulation at time t_o. Figure 4.7a shows an example of the first-order hybrid analysis applied to the client–server model, Model A with the switching behaviour shown in Figure 4.1a.



(a) First-order analysis: means

(b) First-order analysis: variance

(c) Second-order analysis: variance

Figure 4.7: Hybrid analysis approximation of the mean and variance of populations in the client–server model. The simulation replaces ODEs in the interval $[1.8, 2.4]$. The dotted lines are obtained from simulation and the dashed lines are the approximations from ODEs.

It can be seen that the hybrid analysis improves accuracy of the ODE approximations of means. However, since the instances of simulation always get started at times t_s with deterministic initial populations, the variances at times t_s are equal to 0. Figure 4.7b demonstrates this problem, addressed by the second-order hybrid analysis below.

4.3.2 Second-order hybrid analysis

The *second-order hybrid analysis* restarts the simulation using, in addition to the ODEs for means, the ODEs for covariances of populations. The modification produces more accurate estimates of mean populations and covariances $\widehat{\mathbb{E}}^2[\cdot]$. At each time t_s where simulation replaces the ODEs, each replication of the simulation is started with populations drawn from a multivariate normal distribution with mean and covariance matrix given by the respective values from the solution to the ODEs at time t_s. There are several technical issues that have to be considered.

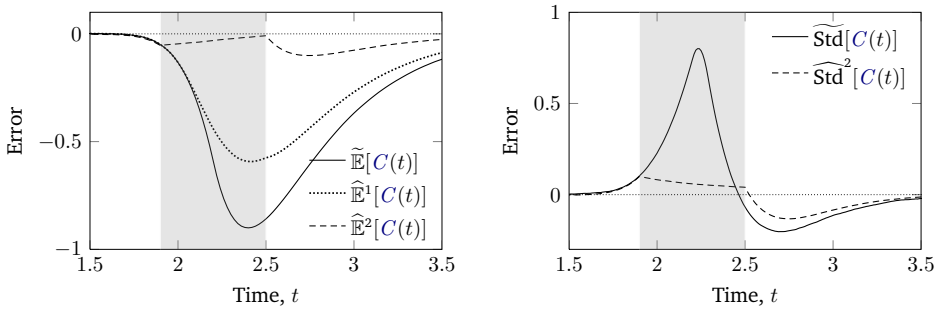
First, in order to simulate a multivariate normal distribution, the covariance matrix has to be positive definite. However, approximations from the ODEs at times t_s may not necessarily be positive definite. Therefore, we have implemented a sampling method of Wang and Liu

[188] which transforms the covariance matrices to have positive real eigenvalues (necessary and sufficient condition for the matrix to be positive definite).

Second, the support of a multivariate normal distribution is the whole of $\mathbb{R}$, whereas the agent populations can only be non-negative. We have implemented a simple greedy method that transforms a set of multivariate normal samples so that they are all positive while trying to maintain the same mean and variances.

Figure 4.7c shows an example of variance approximation for the client–server model. In the ODEs for moments of populations, only ODEs for the second-order moments depend on means and not vice-versa. Therefore, the approximations of means do not take covariances into account. One way of looking at the second-order hybrid analysis is that it feeds back second-order moments into the mean approximations through re-sampling at the times t_s. Intuitively, this should result in an improved accuracy of means from the second-order hybrid analysis over the approximation from the first-order analysis.

Figure 4.8 looks at the error in means and standard deviations of the two hybrid analyses and the original ODE analysis. It plots the difference $\mathbb{E}[\cdot] - \widehat{\mathbb{E}}[\cdot]$ for both analyses and the difference $\mathrm{Std}[\cdot] - \widehat{\mathrm{Std}}^2[\cdot]$ for the second-order hybrid analysis. The first-order hybrid analysis slightly reduces the error of mean approximation. The improvement provided by the second-order hybrid analysis is more significant. The mean approximation error is an order of magnitude smaller, due to the fact that the approximation uses second-order information.



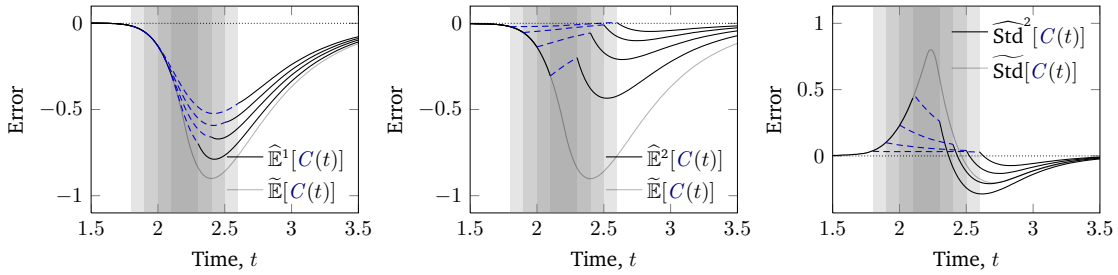
(a) Error: mean (b) Error: standard deviation

Figure 4.8: Error of the hybrid analysis of client–server model (Model A) – the difference between the means, (a), and standard deviations, (b), from simulations and the approximations from each respective method.

4.3.3 Effects of interval length

Intuitively, increasing the length of the time interval where simulations replace the ODEs should increase the accuracy of hybrid approximations of population means. In Figure 4.9 we vary the length of time interval around the switch point in the client–server model and look at the error of first- and second-order hybrid analyses. The length of the simulation interval provides a trade-off between the computational cost and accuracy of the hybrid analysis. In practice, the cost of simulation is significantly higher than the cost of numerically solving the moment ODEs. Therefore the applicability of a hybrid analysis depends on the switch-point distance dynamics. If the model maintains a small switch-point distance for most of the time, the hybrid analyses

provide only a small improvement over stochastic simulation. However, the main motivation of this section was to demonstrate the improvement in accuracy of mean approximations when a second-order information is used. This is further explored in the following section.



(a) First-order: mean (b) Second-order: mean (c) Second-order: st. dev.

Figure 4.9: Effect of the simulation length interval in hybrid analyses on the error of mean agent populations. The dashed segments of the plots show the intervals used in the hybrid analyses. Intervals $[2.1, 2.3], [2.0, 2.4], [1.9, 2.5], [1.8, 2.6]$ were used. The plot without any dashed segment shows the error of the pure ODE analysis.

4.4 Normal moment closure for minimum rates

In case of PCTMC rates with the minimum function, the ODE analysis method from Section 3.4 produces differential equations where the right-hand sides for moments depend only on moments of the same or lower orders. In particular, the equations for means do not depend on variances and covariances of populations. As a consequence, the approximation of terms with minimum function is subject to error as investigated in Section 4.2.

The hybrid simulation introduced in Section 4.3 incorporates variance information into the approximation for means. However, this is at the expense of an increased computational cost. A better approach would be to replace the expectations of minimum terms with an expression that would depend on higher-order moments, similar to the moment closures for polynomial rates described in Section 3.4.2.

In this section, we extend the moment closure framework to rates that incorporate the minimum function. Using the the result in Theorem 4, we assume that the populations in a PCTMC are approximately distributed according to a normal random variables.

The *min-normal moment closure* aims to improve the approximation of expectations such as $\mathbb{E}[\min(X_i(t), X_j(t))]$, often arising on right-hand sides of moment ODEs for PCTMC models coming from PEPA process algebra or stochastic Petri nets. The mean-field closure, using the approximation $\min(\mathbb{E}[X_i(t)], \mathbb{E}[X_j(t)])$, is accurate when the switch-point distance is small as shown in Section 4.2, that is the time intervals when the two means $\mathbb{E}[X_i(t)]$ and $\mathbb{E}[X_j(t)]$ are sufficiently distant. This depends on the variance of the two random variables and large errors occur whenever $\mathbb{E}[X_i(t)] \approx \mathbb{E}[X_j(t)]$.

We can produce a better estimate for the minimum expression under the normal assumption. The hybrid analysis from the previous section has shown that incorporating variance information into the analysis of means can improve accuracy of the ODE analysis. Using a result for the moments

of a minimum of two bivariate normal random variables [47], we can obtain the following identity for (A, B) bivariate normal:

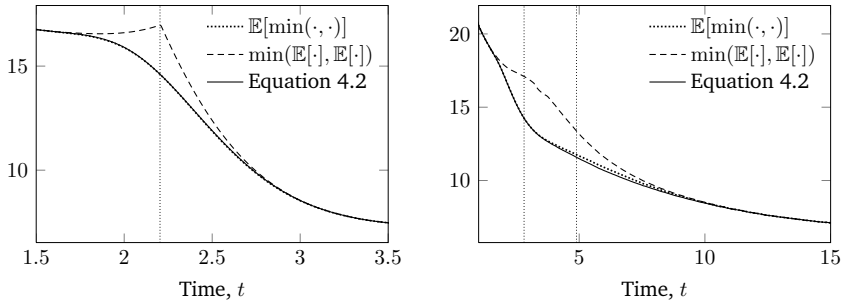
$$\mathbb{E}[\min(A,B)] = \mathbb{E}[A] \cdot \Phi\left(\frac{\mathbb{E}[B]-\mathbb{E}[A]}{\theta}\right) + \mathbb{E}[B] \cdot \Phi\left(\frac{\mathbb{E}[A]-\mathbb{E}[B]}{\theta}\right) - \theta \cdot \phi\left(\frac{\mathbb{E}[B]-\mathbb{E}[A]}{\theta}\right) \tag{4.2}$$

where $\theta = (\text{Var}[A] - 2\text{Cov}[A,B] + \text{Var}[B])^{1/2}$ and Φ and ϕ are the CDF and PDF of a standard normal random variable. The expression $\mathbb{E}[A] - \mathbb{E}[B]$ can be seen as the expectation of a switch-point distance and the variable θ as its standard deviation. Because A, B are normally distributed, the switch-point distance is also normally distributed. Intuitively, Equation 4.2 weighs the contribution of the two arguments of a minimum function to its expectation by the probabilities of the switch-point distance being positive or negative respectively.

The right-hand side of a second-order moment ODE contains terms of the form $\mathbb{E}[C \cdot \min(A,B)]$. In that case, our experiments suggest that a good heuristic is to insert C into the above equation in a way that captures some of the dependency between the 3 variables:

$$\mathbb{E}[C \cdot \min(A,B)] \approx \mathbb{E}[CA] \cdot \Phi\left(\frac{\mathbb{E}[CB]-\mathbb{E}[CA]}{\theta \cdot C}\right) + \mathbb{E}[CB] \cdot \Phi\left(\frac{\mathbb{E}[CA]-\mathbb{E}[CB]}{\theta \cdot C}\right) - E[C] \cdot \theta \cdot \phi\left(\frac{\mathbb{E}[CB]-\mathbb{E}[CA]}{\theta \cdot C}\right) \tag{4.3}$$

Figure 4.10 demonstrates the potential improvement provided by this closure. It shows the exact expectation $\mathbb{E}[\min(C(t), S(t))]$ and the difference between the mean-field approximation and the expression from Equation 4.2. The approximation no longer shows large error when the switch-point distance approaches zero. As expected, an error is still present due to the fact that the populations are not normally distributed.



(a) Model A (b) Model B

Figure 4.10: Difference between approximations of the expectation of rate $\min(C(t), S(t))$ in the client–server model. All expectations are obtained from simulation.

Using Equation 4.2, we can replace all occurrences of the minimum function on the right-hand side of the second-order moment ODEs for a PCTMC model. The general structure of the resulting system of ODEs can be seen in Figure 4.11. Mean ODEs require second-order moments to calculate the variable θ above. Otherwise, moment ODEs are closed at their particular order, that is the ODEs for n^{th}-order moments only require moments of order up-to and including n.

Unlike in the case of normal closure for polynomial rates, higher-order moment ODEs generated by the min-normal closure do not depend on moments of even higher orders and therefore the provided approximation is the same, regardless of the maximum order n.

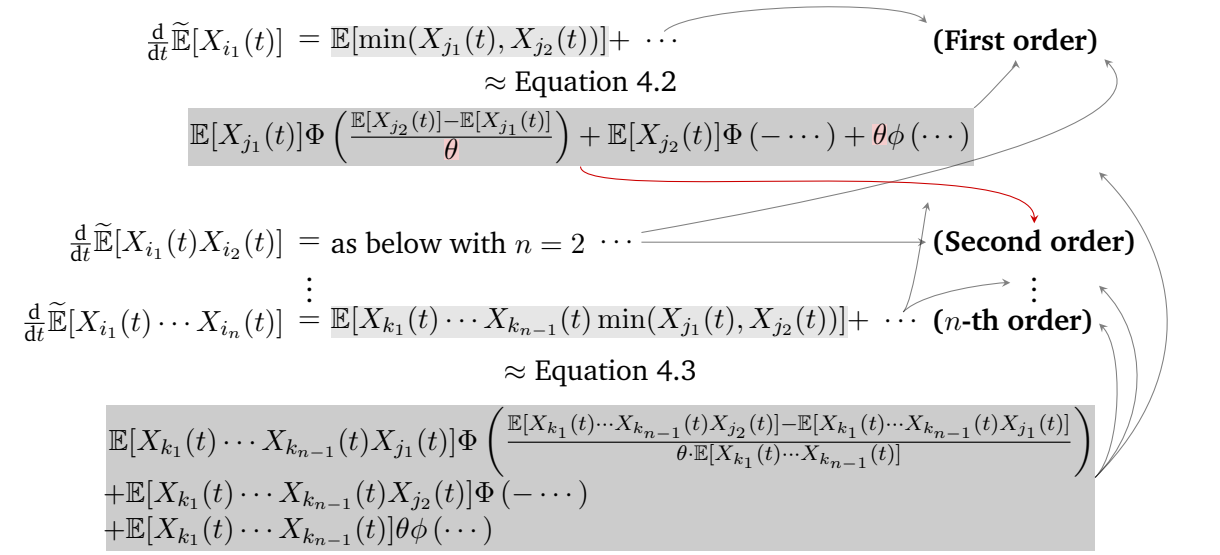


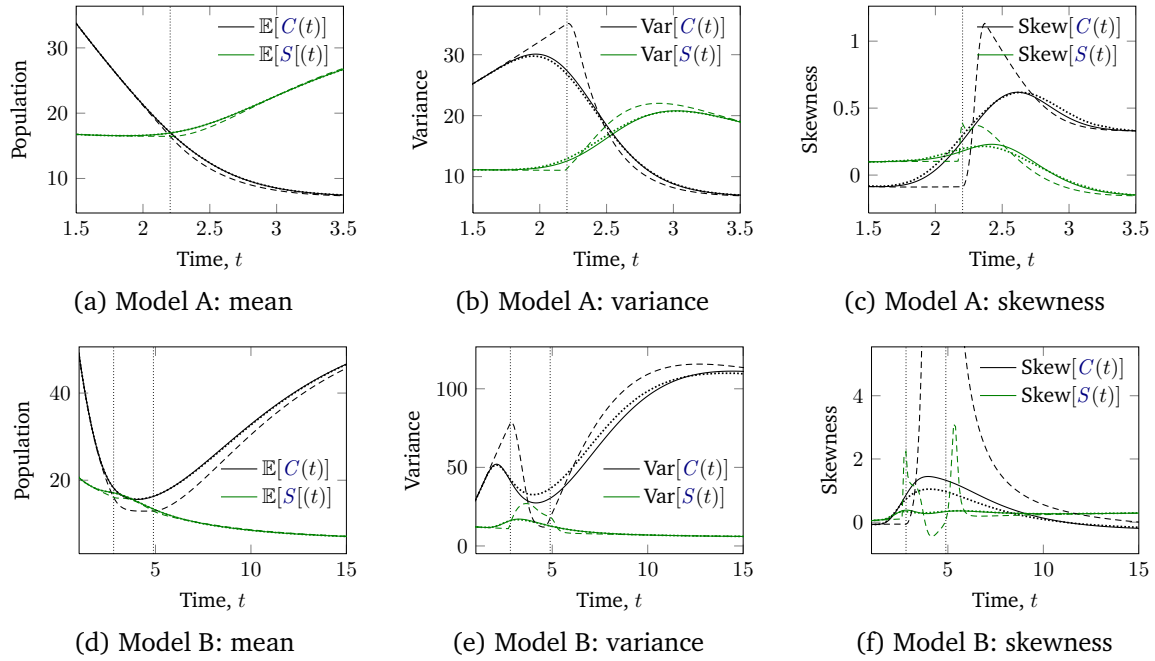
Figure 4.11: Structure of moment ODEs closed by the min-normal closure.

Figure 4.12 shows the improved approximation provided by the min-closure on the two versions of the client–server model from Section 4.2.1. The error in all three shown moments, mean, variance and skewness, is significantly reduced, especially at times with small switch-point distance. However, the error of the min-closure approximation is still affected by the switch-point distance. This is due to the assumption of normally distributed populations and also due to the additional approximation introduced by Equation 4.3.

Theorem 3 no longer formally applies to ODEs closed under the min-closure. However, in combination with Theorem 4, it gives an intuitive justification that the error of approximation decreases as the system scale increases. In this case, the system of ODEs is no longer invariant under scaling – this is caused by the presence of the normal CDF and PDF on the right-hand side of the equations. Figure 4.13 demonstrates the possible convergence and compares the error at different scales to the original approximation. As expected, the error is significantly smaller and comparable to that of the second-order hybrid simulation from Figure 4.8.

4.5 Closure comparison

In this section we numerically compare the approximations from different moment closures for two PCTMC models under a large number of parameter regimes. This will give greater confidence in the use of moment closures based on the normal distribution and justify their use in the following chapters of this thesis.



(a) Model A: mean (b) Model A: variance (c) Model A: skewness

(d) Model B: mean (e) Model B: variance (f) Model B: skewness

Figure 4.12: Moments of populations in the client–server model, Model A The solid lines are approximations by the ODE analysis and dotted lines estimates from 10^8 replications of stochastic simulation. The dashed lines are the respective approximations from the original moment equations, Figure 4.2.

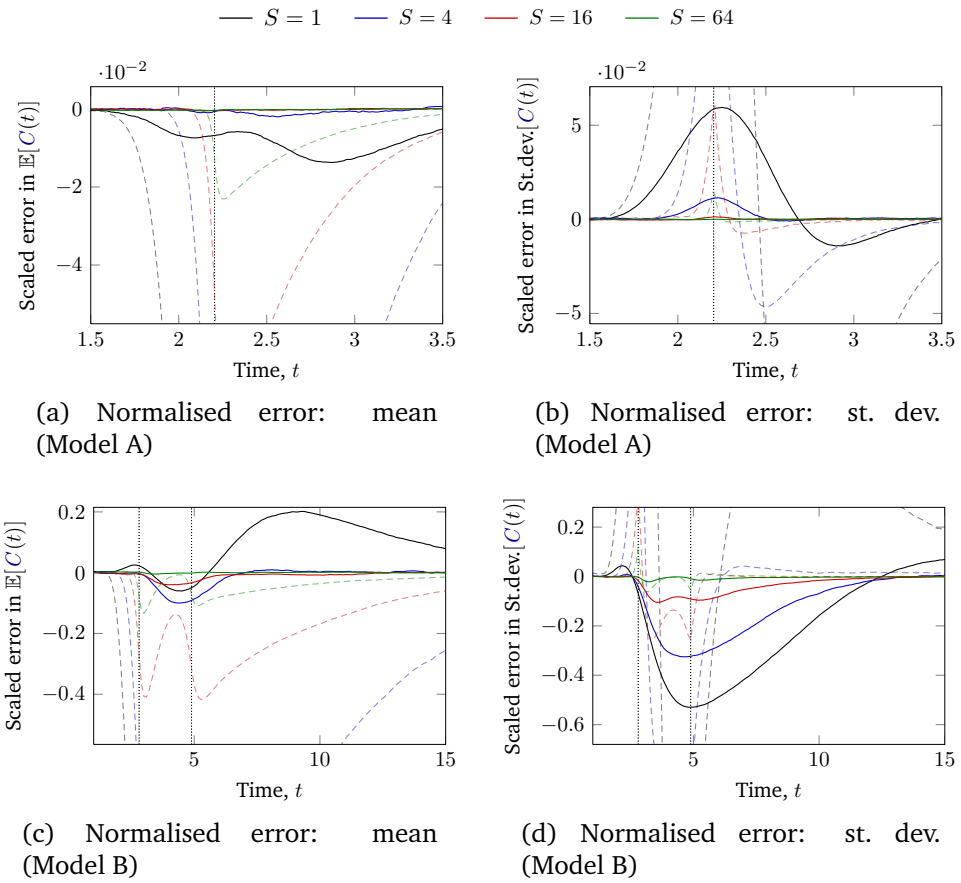
4.5.1 Evaluation framework

We evaluate the error of available ODE approximations for the client–server model and the peer-to-peer model from Section 3.2.1. The rates in the client–server model are linear and contain minimum functions – we compare the GPEPA closure and the min-closure from Section 4.4. The peer-to-peer model contains linear and quadratic rates. We compare the mean-field approximation and ODE approximations closed at second and third orders respectively.

For each model, we numerically compute the approximations for a large number of parameter combinations. For the client–server model, we start with Model A and Model B and vary the number of servers, the server break rate and data transfer rate giving a total of 500 combinations. Because of the size of the models, an exact computation of moments is not feasible. We use an improved version of the stochastic simulation algorithm from Section 3.3 which also computes confidence intervals for means and variances. For each parameter combination, we record the maximum and average error of each approximation. Table 4.2 gives an overview. In the paper by Guenther et al. [3] we have additionally evaluated the closures on a spatial process algebra model of routing in wireless sensor networks where the underlying PCTMC contains third-order rate function and also evaluated the accuracy of a moment closure based on the log-normal distribution.

Simulation with confidence intervals

In order to allow a fair error comparison between simulations and ODE approximations, we implemented a PCTMC simulation algorithm, which generates independent traces until a certain confidence interval is reached for all the population moments that we are estimating. The



(a) Normalised error: mean (Model A)

(b) Normalised error: st. dev. (Model A)

(c) Normalised error: mean (Model B)

(d) Normalised error: st. dev. (Model B)

Figure 4.13: Effects of scaling of the client–server model on the normalised error around the switch point events. The solid lines are approximations by the ODE analysis and the dashed lines are the respective errors from the original moment equations Figure 4.3.

confidence interval for the sample statistics is computed using a Student's t-distribution with the degrees of freedom depending on the sample size. In order to compute the confidence interval of variance estimates, we also keep track of the third- and fourth-order central sample moments. As an example, in the client–server Model A we say that the simulation estimates of mean and variance of the client population have converged if the relative half-width of the 95% confidence interval is $< 1\%$ at any point in time. To achieve this, the model requires about 10^6 simulation traces. Some parameter combinations require even more replications. We noticed that even small parameter changes in some models can heavily impact the convergence behaviour of the accurate simulation. Overall, the simulations used in this section required several CPU days on a standard desktop computer.

Table 4.2: An overview of the models used to compare the different closures. The third column shows the number N of different populations in the model. The last column shows the number of different parameter configurations we evaluated the models on.

	Rates	**Closures**	N	$\#P$
Peer-to-peer	linear, quadratic	Mean-field		
		Normal order 2, 3	5	300
Client–server	linear, $\min$ of linear	Min, min-normal	6	500

Computation of error

To evaluate the accuracy of different closures we compute population moments in both the models above, under a large number of parameter configurations. For a particular model and a set of parameters, the simulation provides a confidence interval estimate

$$[\mathbb{E}[h(\boldsymbol{X}(t))]^L, \mathbb{E}[h(\boldsymbol{X}(t))]^U]$$

of each moment specified by $h(\boldsymbol{X}(t))$ at each time point t until a specified time t_f. At the same time, each closure provides an approximation $\widetilde{\mathbb{E}}[h(\boldsymbol{X}(t))]$. The absolute error of the closure for the moment $h(\boldsymbol{X}(t))$ at time t is

$$e_{\text{abs}}(h(\boldsymbol{X}(t))) = \begin{cases} 0 & \text{if } \mathbb{E}[h(\boldsymbol{X}(t))]^L \leq \widetilde{\mathbb{E}}[h(\boldsymbol{X}(t))] \leq \mathbb{E}[h(\boldsymbol{X}(t))]^U] \\ \widetilde{\mathbb{E}}[h(\boldsymbol{X}(t))] - \mathbb{E}[h(\boldsymbol{X}(t))]^U & \text{if } \widetilde{\mathbb{E}}[h(\boldsymbol{X}(t))] > \mathbb{E}[h(\boldsymbol{X}(t))]^U \\ \mathbb{E}[h(\boldsymbol{X}(t))]^L - \widetilde{\mathbb{E}}[h(\boldsymbol{X}(t))] & \text{if } \widetilde{\mathbb{E}}[h(\boldsymbol{X}(t))] < \mathbb{E}[h(\boldsymbol{X}(t))]^L \end{cases} \tag{4.4}$$

To express the relative error, we divide the absolute error by the point estimate, i.e.

$$e_{\text{rel}}(h(\boldsymbol{X}(t))) = e_{\text{abs}}(h(\boldsymbol{X}(t))) \cdot \frac{2}{|\mathbb{E}[h(\boldsymbol{X}(t))]^L + \mathbb{E}[h(\boldsymbol{X}(t))]^U|}$$

For each model, we look at means and standard deviations of all populations when available. We aggregate the respective errors at each order: For each time t, we define the average/maximum first-order error as the average/maximum relative error across all means, that is

$$e^1_{\text{avg}}(t) = 1/N \sum_i^N e_{\text{rel}}(X_i(t)) \qquad\qquad e^1_{\text{max}}(t) = \max_{i=1,\ldots,N} e_{\text{rel}}(X_i(t))$$

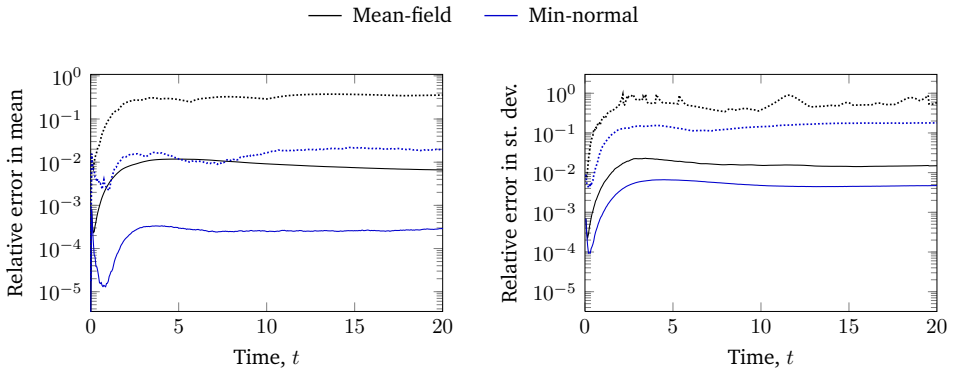
Similarly, we define the second-order aggregate errors $e^2_{\text{avg}}(t)$ and $e^2_{\text{max}}(t)$ by replacing $\mathbb{E}[h(\boldsymbol{X}(t))]$ above with $\text{Std}[\boldsymbol{X}(t)]$. For each closure, we further aggregate the above errors by taking the average/maximum of each error across a large number of parameter combinations. We define $\bar{e}^i_{\text{avg}}(t)$ and $\bar{e}^i_{\text{max}}(t)$ as the average of $e^i_{\text{avg}}(t)$ and maximum of $e^i_{\text{max}}(t)$ over all parameter combinations respectively. Additionally, we also look at the effects of scaling initial populations on the error of the moment closure approximations. We pick a single parameter configuration and calculate the aggregate average and maximum errors $e^i_{\text{max}}(t)$ and $e^i_{\text{avg}}(t)$. We repeat this when the initial populations in the model are multiplied by a scale constant $S \in \{1, 10, 100\}$.

4.5.2 Numerical results

In this section we evaluate accuracy of the different moment closures with respect to results from the accurate simulation. For each of the two example models, we plot $\bar{e}^i_{\text{avg}}(t)$ and $\bar{e}^i_{\text{max}}(t)$ for $i = 1, 2$, the relative errors in mean and standard deviation, aggregated over all parameter combinations. Additionally, we plot $e^i_{\text{avg}}(t)$ and $e^i_{\text{max}}(t)$ for a single parameter combination at 3 different scales of the system, illustrating the improved accuracy as the model size increases. Table 4.3 compares the numerical values of these errors.

Client–server model

Figure 4.14 shows the average and maximum relative errors for the client–server model. The mean-field mean approximations are quite accurate, with maximum error 37.7% and average error no more than 2.3%. The min-normal closure is particularly effective here and brings the errors down to 2.48% and 0.1% respectively. Although in many cases the approximation of standard deviation is qualitatively accurate, the maximum error is quite large at 95.2%, with an average at 4%. The min-normal closure results in an improvement in maximum and average error going to 20% and 1.3% respectively.



(a) Mean (b) Standard deviation

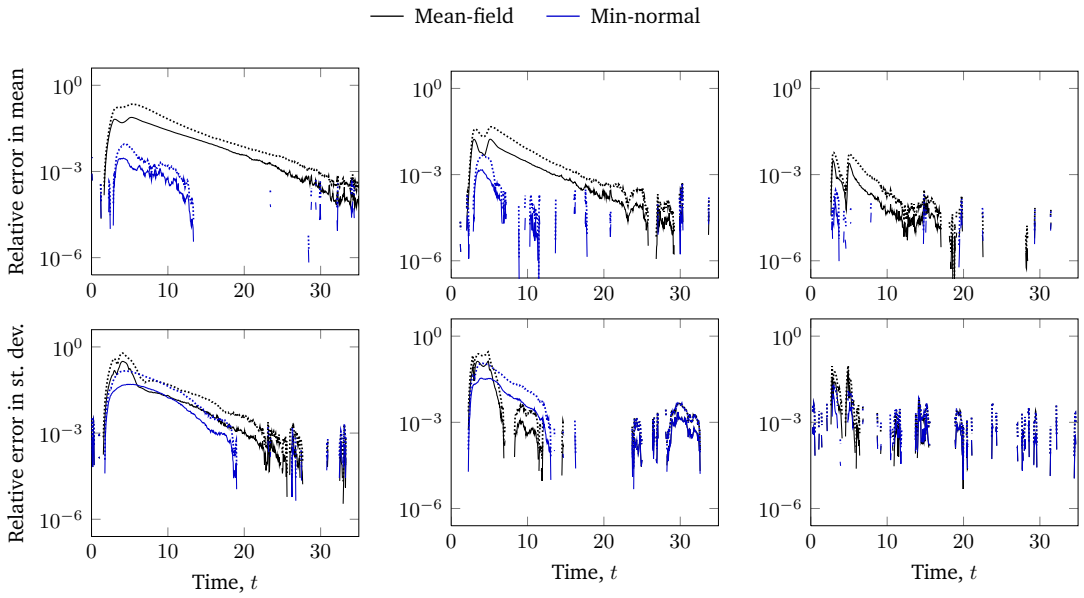
Figure 4.14: Comparison of mean-field and min-normal closures for the client–server model. The solid lines show the average relative error $\bar{e}^i_{\text{avg}}(t)$ and the dotted lines the maximum relative error $\bar{e}^i_{\text{max}}(t)$. The errors are aggregated over a large number of parameters and all the populations.

Figure 4.15 shows the effect of scaling in the client–server model. We can see a decrease in both errors as the model size increases. There are more regions where the errors stay non-zero even at the highest scale. This is possibly caused by the presence of switch points where the minimum approximations are particularly inaccurate.

Hybrid peer-to-peer model

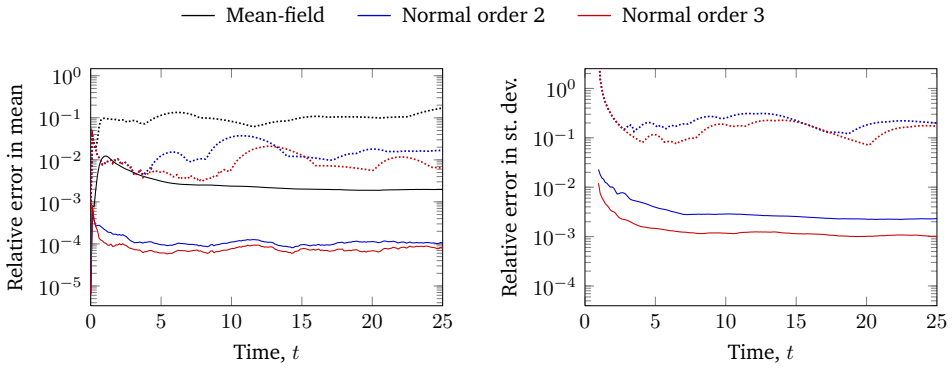
Figure 4.16 shows the average and maximum relative errors $\bar{e}^i_{\text{avg}}(t)$ and $\bar{e}^i_{\text{max}}(t)$ in the peer-to-peer model.

In case of approximations of population means, the mean-field analysis gives quite accurate results, with the average error over across populations in the order of 1.2% and the maximum error of 17% occurring only in certain populations and limited time intervals for each parameter configuration. As we use higher-order moments, we can see the error decrease. The second-order normal closure improves these to 0.08% average and 3.7% maximum error respectively and the third-order further to 0.07% and 2%. The normal closures give quite accurate approximations to standard deviations. For a short initial time period, the relative error is higher due to very small values of the standard deviation. However, for most of the considered time, the second-order normal closure gives a maximum error of around 30% and average error 2% and the third-order closure reduces these to 12% and 0.6% respectively. Figure 4.17 shows the relative errors for a single parameter combination at 3 different scales of the system – when initial populations are scaled by 1, 10 and 100 respectively.



(a) $S = 1$ (b) $S = 10$ (c) $S = 100$

Figure 4.15: Effects of scaling on the accuracy of moment closures in the client–server model. The solid lines show the average relative error $e^i_{\text{avg}}(t)$ and the dotted lines the maximum relative error $e^i_{\text{max}}(t)$. Gaps on the plots represent zero values of the error.



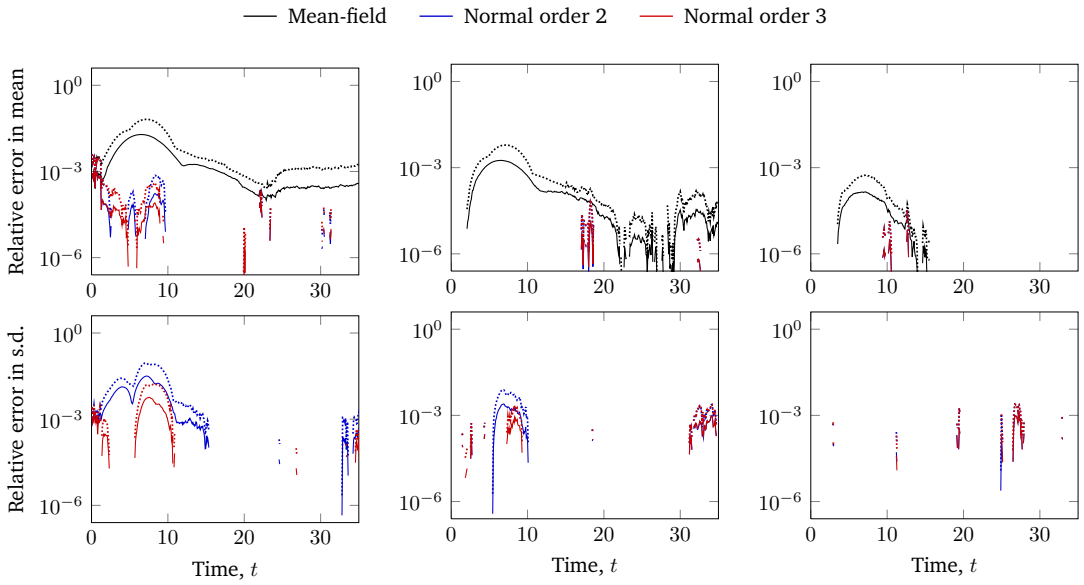
(a) Mean (b) Standard deviation

Figure 4.16: Comparison of closures for the peer-to-peer model. The dotted lines are maximum relative errors $\bar{e}^i_{\text{max}}(t)$ over all the experiments, the solid lines the average relative errors across a number of different model parameters $\bar{e}^i_{\text{avg}}(t)$.

We can see that the error in all the 3 closures decreases with higher scales, both in case of means and standard deviations. At the scale $S = 100$, the normal closures give a zero error with respect to the 2% confidence interval estimate from the simulation for most of the time.

4.6 Conclusion

In this chapter we made several attempts to improve the accuracy of ODE analysis of PCTMC models. We introduced new techniques that allow us to incorporate variance information into mean approximations for models with minimum rates. The first technique uses the standard deviation of a switch-point distance to predict where a large error is likely. The second-order hybrid analysis replaces ODEs with stochastic simulation in such regions and uses approximate



(a) $S = 1$ (b) $S = 10$ (c) $S = 100$

Figure 4.17: Effect of scaling on the accuracy of moment closures in the peer-to-peer model. All plots are shown with the same (logarithmic) scale. The gaps on the plots represent zero error values.

Table 4.3: Summary of the aggregate relative (%) error in the benchmark models. The numbers on the left of each column are the maximum of $\bar{e}^i_{\text{max}}(t)$ over all t and the numbers on the right the maximum of $\bar{e}^i_{\text{avg}}(t)$ respectively.

	Mean-field				**(Min-)normal 2**				**Normal 3**			
	mean		s.d.		mean		s.d.		mean		s.d.	
	max	avg	max	avg	max	avg	max	avg	max	avg	max	avg
Peer-to-peer	17	1.2	—		3.7	.08	30	2	2	.07	12	.6
client–server	37.7	2.3	95.2	4.0	2.48	0.1	20	1.3	—			

the covariance matrix for more accurate initial sampling of the simulation. This quantitatively improves the error at a large computational cost. Finally, we present a new moment closure which approximates the expectation of minimums using an expression for the minimum of bivariate normal random variables, with covariance matrix obtained from second-order ODEs. This leads to an approximation with comparable accuracy to the second-order hybrid analysis but much lower computation cost.

We have validated the new closure as well as the second- and third-order normal closures on two sample models under a large number of different parameter configurations. The results give us confidence in applying these closures. This is a crucial development for the remainder of this thesis. In the following chapter, we extend the moment ODEs with additional equations that capture moments of accumulated rewards. These equations depend on population ODEs and also require analogous closure approximations. The improved understanding from this chapter helps us to better assess the error of the extended reward ODE analysis. Moreover, in Chapter 6, we show another use of the minimum function in a new class of PCTMC models that use feedback for transient control of rewards. We will show that the minimum closure is crucial in obtaining a usable approximation of second- and higher-order moments of populations in such models.

Chapter 5

PCTMCs with accumulated rewards

Key contributions		
Extend ODE analysis to moments of accumulated rewards	5.3	[9]
Use moments to compute distributions of completion times	5.3.4	[9]
Propose an optimisation problem for efficient evaluation of energy–performance trade-offs	5.5	[9, 10]

5.1 Introduction

Energy consumption is a critical factor in practical operation of massive computer systems. Whether in wireless networks, virtualised services that run on data centres, energy consumption, temperature and total operational costs of the system have to be taken into account as justified in Chapter 1. Traditionally, such metrics are captured as rewards in a stochastic model representing the system. However, given the scale of a typical data centre, it would be impractical to perform traditional reward analysis on these massively parallel systems. The ODE analysis techniques from Section 3.4 make it possible to analyse systems which exhibit a high degree of parallelism at a much lower cost than stochastic simulation. Additionally, as demonstrated in Chapter 4, the accuracy of the ODE analysis techniques can be maintained at a similar level to simulation.

The aim of this chapter is to extend the ODE analysis of PCTMCs to compute accumulated rewards (reviewed in Section 2.3). We show that many measures can be constructed precisely as functions of existing population moments, while for others, we derive additional differential equations augmenting the ODE analysis. The main contributions of this chapter are:

Impulse rewards In Section 5.2.2 we show how impulse rewards can be obtained via simple addition of auxiliary action counting populations into PCTMC models. This allows the ODE analysis to be used to compute moments of impulse rewards and composite rewards that are combinations of population based expressions, impulse rewards and accumulated rewards below.

Continuously accumulated rewards In Section 5.3.1 we describe how to represent means of continuously accumulated reward measures as ODEs that augment the set of ODEs for population means. We show that higher moments of reward measures can similarly be generated. This gives access to rewards in substantially larger Markov models than has previously been possible, for example in the approach of Telek and Rácz [175], who analysed Markov Reward Models of the order of 10^6 states.

Completion times We demonstrate how reward passage times, or so-called *completion times*, can be constructed and expressed in terms of reward moments, giving for instance the distribution of the time until the accumulated energy consumption of a server reaches a certain level.

Trade-off between energy consumption and performance Finally, we illustrate the technique on a model of energy consumption in a client–server system with server failures and hibernation. We show how the reward metrics and previously derived passage time measures can form a constrained global optimisation problem that captures the trade-off between energy consumption and service level agreement (SLA) compliance. We demonstrate how the efficiency of ODE analysis allows us to tackle such problems at a reasonable computational cost.

In addition, these contributions are reflected in features of the GPA tool, described in detail in Chapter 8. We implemented an extension to the syntax of GPEPA to allow a convenient inclusion of impulse rewards based on action counts. We implemented the ODEs for moments of accumulated rewards in general PCTMC models. We integrated the framework of Tari et al. [173] to compute moment-based distribution bounds that can be used to get tighter global passage time estimates, Section 3.5.4, and completion time estimates, Section 5.3.4. We extended the simulation algorithm to be able to compute estimates of accumulated rewards. Finally, we provide a mechanism to explore large optimisation problems such as the energy–performance trade-off; both explicit parameter exploration and approximate global optimisation algorithms are available. All numerical examples in this chapter were produced with the GPA tool.

5.1.1 Accumulated rewards

Consider the client–server model described in Section 3.2.3. In Chapter 3, we have summarised how the ODE analysis can be used to efficiently and accurately approximate means and higher-order moments of populations in PCTMCs. These are useful in gaining an understanding of the dynamics of the system from both qualitative and quantitative point of view. However, modellers are often interested in further quantities influenced by the system behaviour. Many of them can be thought of as "rewards" generated by the system. Some rewards can be expressed as expressions of populations at a desired time instant. However, a large number of useful rewards take into account the whole history of the population process. In this chapter, we are interested in *accumulated rewards*, quantities which are accumulated over time.

Impulse rewards

The so-called *impulse rewards* accumulate rewards discretely with system transitions. For example, in the client–server model, the service provider can ask for a fixed fee each time a client successfully processes its acquired data. In GPEPA, this can be captured by so-called *action-counting* processes [103] which keep track of the number of times an action fired until time t . For example, the income from the client fees can be expressed in terms of the process $X_{think}(t)$ that counts the number of fired $think$ actions in the system until time t

$$\mathcal{A}_{\text{fee}}(t) = c_{\text{fee}} \cdot X_{think}(t).$$

Continuously accumulated rewards

One example of a reward which grows continuously over time in the client–server model is the total energy consumption of all servers in the model. We can assume that each server consumes energy at a rate specific to one of its three states, r_S, r_{S_g}, r_{S_b} for the idle, serving and broken states respectively. This is a reasonable assumption that can be confirmed by experiments. Figure 5.1 shows an example of energy consumption and CPU utilisation relationship for two desktop computers with 2 and 4 cores respectively, obtained from a series of measurement experiments [7]. The different states of the computers are clearly visible. For example, Computer 1 spends most of the time in 3 distinct states – with 0%, 50% and 100% CPU utilisation (idle, one and two active CPU cores). These could represent the states *Server_broken*, *Server* and *Server_get* respectively. At each of these states, the power consumption is roughly constant, $60W$, $90W$ and $100W$ respectively. These figures are only rough estimates used for illustration purposes and we discuss how a more accurate power model could be constructed in Section 5.6.

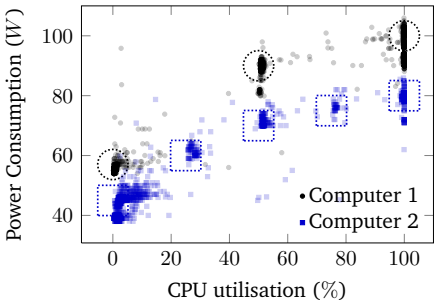


Figure 5.1: Example of accumulation rates corresponding to energy consumption of a server. Two computers were used with 2 and 4 physical CPU cores respectively. The dotted areas show clusters representing the different states the computers spent most of the time in.

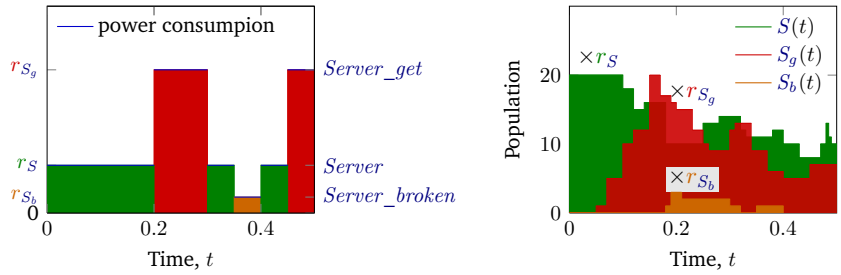
Figure 5.2a shows an example of the energy consumption for a single server. In each state, the server has a constant power consumption. The energy consumed until time t is a linear combination of the time durations the server spent in each individual state, weighted by the respective rate constants r_S, r_{S_g}, r_{S_b}. This can be expressed as the area under the plot in Figure 5.2a. Figure 5.2b shows an example of energy consumption of multiple servers. For each time interval (t_1, t_2) between two successive changes in the population process, the total consumption is

$$(t_2 - t_1)(S(t_1) \cdot r_S + S_g(t_1) \cdot r_{S_g} + S_b(t_1) \cdot r_{S_b}).$$

By summing over all the time intervals during which populations stay constant, we can define the total energy consumption up to time t as an integral over a linear combination of populations:

$$\mathcal{A}_{\text{energy}}(t) = \int_0^t S(u) \cdot r_S + S_g(u) \cdot r_{S_g} + S_b(u) \cdot r_{S_b} \mathrm{d}u \tag{5.1}$$

Finally, we can combine the continuously accumulated rewards and impulse rewards. For example, the main metric of interest to the service provider could be the *total* income from



(a) Energy consumption of a single server

(b) Total server energy consumption

Figure 5.2: Total energy consumption of servers in the client–server model as an accumulated reward expressed as an integral over the server state populations.

running the system. Ignoring further infrastructure costs, this would be

$$\mathcal{A}_{\text{total}}(t) = \mathcal{A}_{\text{fee}}(t) - c_{energy} \cdot \mathcal{A}_{\text{energy}}(t) \tag{5.2}$$

where c_{energy} is a cost for a unit of energy consumption.

Accumulated rewards may grow indefinitely with time and to provide any information about their values in the steady state, the rewards need to be normalised in some way. It is common to look at the linear rate of increase in the reward in the steady state: that is, to look at the limit of the reward divided by t as t goes to infinity.

Measures of interest

We extend the ODE analysis to compute moments of accumulated rewards. Similar to population moments, these can be further used to compute various derived metrics, such as completion time probabilities. The following list summarises all quantities we are interested in. The ones marked by $(*)$ can already be calculated via ODE approximations, whereas ODE approximations of the remaining ones are the subject of investigation in this chapter:

Moments of populations* means and higher-order and joint moments of agent populations at a finite time $t \geq 0$, for example $\mathbb{E}[C(t)]$, $\text{Var}[C(t)]$, $\mathbb{E}[S(t)\,C(t)]$ in the client–server model.

Moments of populations in steady state* mean populations and moments in the *steady state* of the system (as $t \to \infty$).

Passage time distributions* passage-time distribution of individual agents or groups of agents reaching a particular state. For example the time it takes a single client to perform its first $think$ action.

Moments of impulse rewards* mean and higher-order moments of quantities which accumulate in discrete steps after each successful transition from a given transition class in a PCTMC, at a finite time $t \geq 0$, for example $\mathbb{E}[\mathcal{A}_{\text{fee}}(t)]$, $\text{Var}[\mathcal{A}_{\text{fee}}(t)]$.

Moments of continuously accumulated rewards means and higher-order and joint moments of continuously accumulated agent populations and moments of combinations of continuously and impulse accumulated rewards at a finite time $t \geq 0$, for example $\mathbb{E}[\mathcal{A}_{\text{energy}}(t)]$ and $\text{Var}[\mathcal{A}_{\text{total}}(t)]$.

Mean normalised rewards in steady state normalised accumulated quantities (divided by t) and their moments in the steady state of the system (as $t \to \infty$)

Completion-time distributions the passage-time or completion-time distribution for a system to reach a particular reward level. For example the time it takes the reward $\mathcal{A}_{\text{energy}}(t)$ to reach an energy consumption of 100 units.

All of these quantities can be obtained via stochastic simulation of the system. However, with increasing initial populations, simulation becomes expensive due to frequent events of short duration. Moreover, with increasing order of the moments of interest, the number of simulation replications needed greatly increases. For the population based quantities above, Section 3.4 shows how to derive a system of ODEs that approximates the transient evolution of means and higher-order and joint moments of populations. Numerically solving these ODEs is computationally less expensive than simulation and can thus provide fast access to the moments with a reasonable degree of accuracy. Population moments can be further used to derive various passage time measures [4], as described in Section 3.5. The nature of the approximation provided by the moment ODEs is investigated in Chapter 4, proposing a number of heuristics that can improve the accuracy of the ODE analysis. This framework thus gives us access to accurate approximations of the quantities of interest marked by $(*)$.

The main contribution of this chapter is an extension of the ODE analysis of PCTMCs, with ODEs approximating moments of continuously accumulated quantities. We give an exact form of these ODEs, in a theorem analogous to Theorem 1, and show how to apply moment closure approximations to get a system that can be numerically solved alongside the ODEs for moments of populations, giving approximations to moments of rewards such as $\mathcal{A}_{\text{energy}}(t)$ and $\mathcal{A}_{\text{total}}(t)$. We show how to use the moment approximations to derive approximations to distributions of completion times. We show how the ability to simultaneously compute rewards and passage-times in PCTMC models allows us to accurately capture energy performance trade-offs in models of large scale computer systems.

5.2 Accumulated rewards expressed in terms of populations

We start by showing how population moment ODEs can be used to compute two of the metrics above. A well known property of CTMCs can be used to compute the steady-state normalised accumulated rewards. On the other hand, we show how impulse rewards can be converted to auxiliary populations in a modified PCTMC and thus the ODE analysis directly used to approximate the moments.

5.2.1 Steady state normalised rewards

To access moments of accumulated rewards in the steady state of the system, we can use the ODE analysis without any extensions. The rate of increase in the steady state is

$$\frac{1}{t}\int_0^t X_i(u)\mathrm{d}u \text{ as } t \to \infty.$$

For the reward $\mathcal{A}_{\text{energy}}(t)$ we need to evaluate $\mathcal{A}_{\text{energy}}(t)/t$ as $t \to \infty$.

We can express expectations of such quantities in general by using the steady-state limits of the means of individual populations. This is a corollary of a standard property of CTMCs (previously mentioned in the context of rewards for a process algebra by Ding [68]):

Theorem 5 (Norris [150, Theorem 3.8.1])**.** Let $\{\boldsymbol{X}(t) \in \mathbb{Z}_+^N\}_{t\geq 0}$ be an irreducible, positive recurrent Markov process and $\boldsymbol{f}\colon \mathbb{Z}_+^N \to \mathbb{R}$ a bounded function. Then:

$$\mathbb{P}\left(\frac{1}{t}\int_0^t \boldsymbol{f}(\boldsymbol{X}(s))\mathrm{d}s \to \bar{\boldsymbol{f}}\right) = 1 \qquad \text{as } t \to \infty \tag{5.3}$$

where $\bar{\boldsymbol{f}} = \sum_{\boldsymbol{x}\in\mathbb{Z}_+^N} \lambda_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x})$ and $\lambda_{\boldsymbol{x}}$ is the unique invariant distribution.

> Using the theorem, we can directly get the expression for a mean rate of increase of $\mathcal{A}_{\text{energy}}(t)$ in the steady state:
>
> $$\mathbb{E}\left[\mathcal{A}_{\text{energy}}(t)/t\right] = \mathbb{E}[S(t)]\cdot r_S + \mathbb{E}[S_g(t)]\cdot r_{S_g} + \mathbb{E}[S_b(t)]\cdot r_{S_b} \text{ as } t \to \infty.$$

The method from Section 3.4 provides ODEs with solutions $\widetilde{\mathbb{E}}[\cdot]$ approximating the expectations on the right hand side for a finite t. Finding the fixed point solution of these ODEs then gives an approximation of the desired steady state rates of increase of a reward.

We also note that the Theorem 5 implies that the normalised accumulated rewards are deterministic in the steady state limit and therefore all higher-order moments are products of the respective expectations. For example, the variance of these measures is zero.

5.2.2 Impulse rewards

Impulse rewards increase in discrete steps by a constant after a successful transition in a PCTMC. It is straightforward to extend a PCTMC $\boldsymbol{X}(t) \in \mathbb{Z}_+^N$, driven by transition classes $\mathcal{C}$, with auxiliary populations which capture the number of firings of a transition from a class $c \in \mathcal{C}$. We add a dimension to the state space to get $\boldsymbol{X}'(t) \in \mathbb{Z}_+^{N+1}$ and set

$$\boldsymbol{X}'(0) = (X_1(0), \ldots, X_N(0), 0).$$

The new PCTMC consists of the same transition classes except for c, which we modify to increase the population $X_{N+1}(t)$ after completion. That is, we replace c with c' such that $r_{c'} = r_c$ and $\boldsymbol{\delta}_{c'} = \boldsymbol{\delta}_c + \boldsymbol{e}_{N+1}$, where $e_{N+1} \in \mathbb{Z}_+^N$ is a zero vector with a one as the last element.

> For example, the reward $\mathcal{A}_{\text{think}}(t)$ in the client–server model can be computed by adding a population capturing the number of fired $think$ actions. This requires modification of the transition class $(r_3, \boldsymbol{\delta}_3)$ with rate $r_3(\boldsymbol{X}(t)) = -C_t(t) \cdot r_{think} + \min(C_w(t), S_g(t))r_{data}$ and change vector $\boldsymbol{\delta}_3 = (1, 0, -1, 0, 0, 0)$. The rate stays the same and the new change vector is $(1, 0, -1, 0, 0, 0, 1)$.

Using this construction, we can apply the ODE analysis to extract means and higher-order moments of impulse rewards. In the following section, we show how to derive ODEs capturing moments of continuously accumulated rewards such as $\mathcal{A}_{\text{energy}}(t)$. These depend on auxiliary ODEs allowing us to access *mixed* moments specified by a product of an accumulated reward and a population, such as $\mathbb{E}[\mathcal{A}_{\text{energy}}(t)\,C(t)]$. When the population corresponds to an impulse reward, this allows us to calculate higher-order moments of combined rewards such as $\text{Var}[\mathcal{A}_{\text{total}}(t)]$ which requires the expectation $\mathbb{E}[\mathcal{A}_{\text{energy}}(t)\mathcal{A}_{\text{think}}(t)]$.

5.3 Approximating moments of continuously accumulated rewards via ODEs

In this section, we extend the ODE approximation of PCTMCs to include ODEs for moments of accumulated rewards. In case of higher-order moments, we introduce additional *mixed* moments – expectations of a product of populations and accumulated populations. Using the construction for impulse rewards from Section 5.2.2, this allows us to compute moments of linear combination of accumulated and impulse rewards.

5.3.1 Mean ODEs

We show how to derive approximations for means of accumulated rewards at each time $t \geq 0$. For the energy consumption reward, this is $\mathbb{E}[\mathcal{A}_{\text{energy}}(t)]$. Such expectations are differentiable and hence we can construct a new differential equation for the mean of each required accumulated population. To obtain the right-hand side, we note that since the rewards are always bounded and differentiable, we can swap the differentiation and expectation to get

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\left[\int_0^t X_i(u)\mathrm{d}u\right] = \mathbb{E}[X_i(t)]. \tag{5.4}$$

Numerically solving these simultaneously with the ODEs for population means gives an approximation to mean accumulated rewards.

> For our sample rewards we would take ODEs approximating the first-order moments of populations in the client–server model and add the following ODEs to the system:
>
> $$\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}\left[\int_0^t S(u)\mathrm{d}u\right] = \widetilde{\mathbb{E}}[S(t)]$$
> $$\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}\left[\int_0^t S_g(u)\mathrm{d}u\right] = \widetilde{\mathbb{E}}[S_g(t)]$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}\left[\int_0^t S_b(u)\mathrm{d}u\right] = \widetilde{\mathbb{E}}[S_b(t)].$$

The energy consumption reward can be expressed as a linear combination of the solution to the resulting system of ODEs:

$$\mathbb{E}[\mathcal{A}_{\text{energy}}(t)] \approx r_S \cdot \widetilde{\mathbb{E}}\left[\int_0^t S(u)\mathrm{d}u\right] + r_{S_g} \cdot \widetilde{\mathbb{E}}\left[\int_0^t S_g(u)\mathrm{d}u\right] + r_{S_b} \cdot \widetilde{\mathbb{E}}\left[\int_0^t S_b(u)\mathrm{d}u\right]. \tag{5.5}$$

5.3.2 Higher-order moment ODEs

We now look at the general case of higher-order moments of accumulated populations at each time t, to get for example approximations to variances of accumulated rewards, such as $\mathrm{Var}[\mathcal{A}_{\text{energy}}(t)]$ for $t \geq 0$.

First, we define a shorthand for the accumulated population of X_i up to time t as

$$\overline{X_i}(t) = \int_0^t X_i(u)\mathrm{d}u \tag{5.6}$$

where $1 \leq i \leq N$. From now, unless stated otherwise, we assume $1 \leq i, j \leq N$ when referring to $X_i(t)$ and $X_j(t)$.

For simplicity, we first consider second-order moments of accumulated populations and later show how to extend the technique to higher orders. As in the case of mean populations, we can note that the moments are differentiable and bounded and so we can swap the differentiation and expectation and get ODEs of the form:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\left[\overline{X_i}(t)\overline{X_j}(t)\right] = \mathbb{E}[X_i(t)\overline{X_j}(t)] + \mathbb{E}[X_j(t)\overline{X_i}(t)] \tag{5.7}$$

The right-hand side now contains mixed moments of the form $\mathbb{E}[X_i(t)\overline{X_j}(t)]$. We define ODEs governing these. This time, the process $X_i(t)\overline{X_j}(t)$ is not differentiable, due to jumps in the population process $X_i(t)$, so we cannot simply swap the expectation and differentiation. We can look at the derivative of the expectation of a mixed product of a population and an accumulated population $\mathbb{E}[X_i(t)\overline{X_j}(t)]$ from the first principles to arrive at the following theorem:

Theorem 6. For a PCTMC $\boldsymbol{X}(t) \in \mathbb{Z}_+^N$ and $1 \leq i, j \leq N$

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[X_i(t)\overline{X_j}(t)] = \mathbb{E}[f_{X_i}(\boldsymbol{X}(t))\overline{X_j}(t)] + \mathbb{E}[X_i(t)X_j(t)] \tag{5.8}$$

where $f_{X_i}(\boldsymbol{X}(t))$ is defined in Equation 3.3, i.e. the function such that

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\left[X_i(t)\right] = \mathbb{E}[f_{X_i}(\boldsymbol{X}(t))] \tag{5.9}$$

Proof. See the proof of the more general statement in Theorem 7 below, setting $h_0(\boldsymbol{X}(t)) = X_i(t)$ and $h_1(\boldsymbol{X}(t)) = X_j(t)$. □

We need to evaluate each of the two terms on the right-hand side of Equation 5.8 to numerically solve this ODE together with the rest of the system. The second term $\mathbb{E}[X_i(t)X_j(t)]$ has an approximation $\widetilde{\mathbb{E}}[X_i(t)X_j(t)]$ given by one of the moment ODEs. The exact form of the first term depends on the structure of the PCTMC. Similar to ODEs for moments of populations, a closure heuristic is required to obtain a closed system of ODEs. For example, if the PCTMC comes from a split-free GPEPA model, the first term contains, after moving the expectation through summations and multiplications by constants, terms of the form:

$$\mathbb{E}[X_i(t)\overline{X_j}(t)] \text{ and } \mathbb{E}[\min(g(\boldsymbol{X}(t)), h(\boldsymbol{X}(t)))\overline{X_k}(t)]$$

where g, h are piecewise linear functions (i.e. involve only linear combinations and applications of the $\min$ function) of populations. Theorem 6 can be repeatedly used to obtain ODEs of the former terms. For the latter ones, we can apply the approximation of Equation 3.10 repeatedly to get piecewise linear functions involving terms $\mathbb{E}[X_i(t)\overline{X_j}(t)]$. Alternatively, we can modify the min-closure from Equation 4.3. From our experiments the following approximation leads to more accurate results, when the random variable C is an accumulated population as opposed to a population in Equation 4.3 (the variable C is no longer included in the arguments of ϕ and Φ):

$$\begin{aligned}\mathbb{E}[C \cdot \min(A, B)] \approx \mathbb{E}[C \cdot A] \cdot \Phi\left(\frac{\mathbb{E}[B] - \mathbb{E}[A]}{\theta}\right) + \mathbb{E}[C \cdot B] \cdot \Phi\left(\frac{\mathbb{E}[A] - \mathbb{E}[B]}{\theta}\right) \\ - E[C] \cdot \theta \cdot \phi\left(\frac{\mathbb{E}[B] - \mathbb{E}[A]}{\theta}\right)\end{aligned} \tag{5.10}$$

If the model is splitting, we additionally get terms:

$$\mathbb{E}[f(\boldsymbol{X}(t))\overline{X_j}(t)]$$

where f is a rational function of populations. We can use the approximation from Equation 3.10 to get rational functions involving terms $\mathbb{E}[X_i(t)\overline{X_j}(t)]$. This process is fully automated in the GPA tool described in Chapter 8.

Figure 5.3 shows the general structure of the complete system of ODEs after applying one of the approximating closures.

> We use Theorem 6 and the approximation from Equation 3.10 to find second-order moments of accumulated server populations and thus compute the variance of $\mathcal{A}_{\text{energy}}(t)$. For simplicity, we assume that power consumption of a broken server r_{S_b} is zero. We have:
>
> $$\text{Var}[\mathcal{A}_{\text{energy}}(t)] = r_S^2\text{Var}[\overline{S}(t)] + r_{S_g}^2\text{Var}[\overline{S_g}(t)] + 2 \cdot r_S r_{S_g}\text{Cov}[\overline{S}(t), \overline{S_g}(t)]$$
>
> and also:
>
> $$\begin{aligned}\text{Var}[\overline{S}(t)] &= \mathbb{E}[(\overline{S}(t))^2] - \mathbb{E}[\overline{S}(t)]^2 \\ \text{Var}[\overline{S_g}(t)] &= \mathbb{E}[(\overline{S_g}(t))^2] - \mathbb{E}[\overline{S_g}(t)]^2 \\ \text{Cov}[\overline{S}(t), \overline{S_g}(t)] &= \mathbb{E}[\overline{S}(t)\overline{S_g}(t)] - \mathbb{E}[\overline{S}(t)]\mathbb{E}[\overline{S_g}(t)]\end{aligned}$$

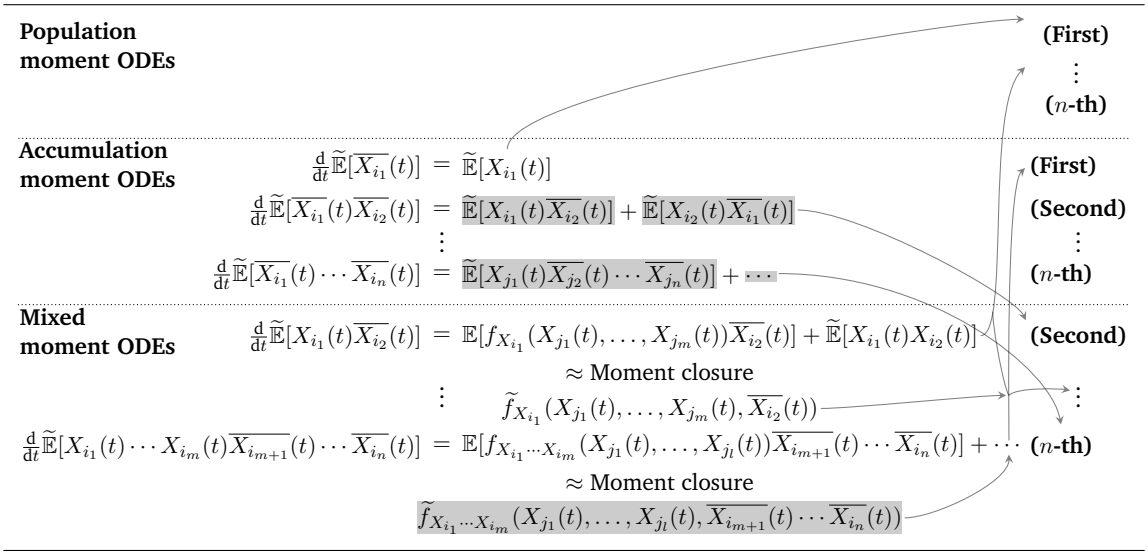


Figure 5.3: Structure of the ODE system approximating moments of populations, accumulations and a mixed product of the two. The arrows show interdependencies between the different groups of ODEs. Functions $\widetilde{f}_{\boldsymbol{x}'}$ are respective transformations of the functions $f_{\boldsymbol{x}'}$ under the chosen moment closure.

We can construct additional ODEs approximating the terms on the right-hand side of these expressions:

$$\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[(\overline{S}(t))^2] = 2\widetilde{\mathbb{E}}[S(t)\overline{S}(t)]$$
$$\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[(\overline{S_g}(t))^2] = 2\widetilde{\mathbb{E}}[S_g(t)\overline{S_g}(t)]$$
$$\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[\overline{S}(t)\overline{S_g}(t)] = \widetilde{\mathbb{E}}[S(t)\overline{S_g}(t)] + \widetilde{\mathbb{E}}[S_g(t)\overline{S}(t)]$$

To get ODEs for mixed moments such as $\widetilde{\mathbb{E}}[S(t)\overline{S_g}(t)]$, we can use Theorem 6 where:

$$f_S(t) = r_{data} \cdot \min(C_w(t), S_g(t)) + r_{reset} \cdot S_b(t) - r_{request} \cdot \min(C(t), S(t)) - r_{break} \cdot S(t)$$

Therefore the ODE describing the moment $\mathbb{E}[S(t)\overline{S_g}(t)]$ is

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[S(t)\overline{S_g}(t)] = r_{data} \cdot \mathbb{E}[\overline{S_g}(t)\min(C_w(t), S_g(t))] + r_{reset} \cdot \mathbb{E}[S_b(t)\overline{S_g}(t)]$$
$$- r_{request} \cdot \mathbb{E}[\overline{S_g}(t)\min(C(t), S(t))] - r_{break} \cdot \mathbb{E}[S(t)\overline{S_g}(t)]$$

We can apply the same procedure to derive an ODE for the other mixed moment $\mathbb{E}[S_b(t)\overline{S_g}(t)]$. The remaining terms $\mathbb{E}[\overline{S_g}(t)\min(C(t), S(t))]$ and $\mathbb{E}[\overline{S_g}(t)\min(C_w(t), S_g(t))]$ require an approximation similar to the one used for ODEs of second-order moments of populations. For example, using the original approximation for GPEPA rates, Equation 3.10, we get

$$\mathbb{E}[\overline{S_g}(t)\min(C(t), S(t))] \approx \min(\widetilde{\mathbb{E}}[\overline{S_g}(t)C(t)], \widetilde{\mathbb{E}}[\overline{S_g}(t)S(t)])$$

> Mixed moments are also useful in calculating higher-order moments of rewards which combine continuously accumulated rewards and impulse rewards. Let $X_{think}(t)$ be the population counting the number of successful *think* transitions. To compute variance of the total reward, $\mathrm{Var}[\mathcal{A}_{\text{total}}(t)]$, we need the system of ODEs that we used to compute $\mathrm{Var}[\mathcal{A}_{\text{energy}}(t)]$, extended with ODEs for $\mathbb{E}[X_{think}(t)]$, $\mathbb{E}[X_{think}(t)^2]$, all the mixed moments $\mathbb{E}[X_{think}(t)S(t)]$, $\mathbb{E}[X_{think}(t)S_g(t)]$ and $\mathbb{E}[X_{think}(t)S_b(t)]$ and all the joint moments required by the right-hand sides of these equations.

5.3.3 Accumulations of products of populations

So far, we have only looked at accumulated rewards where the reward grows as an integral over a *linear combination* of populations. There are cases where a more general function would be desired. One hypothetical example would be if in the client–server model, there is a cost for the connection bandwidth between clients and servers per unit of time. Depending on the used infrastructure, this cost could be proportional to the number of possible pairs of client–server connections. In that case, the reward would accumulate at a rate given by the *product* of the client and server population at each time, that is we would get

$$\mathcal{A}_{\text{network}}(t) = \int_0^t C(u)S(u)\mathrm{d}u.$$

In this section we show how to generalise the presented accumulated rewards to allow accumulations of products of populations by extending Equation 5.4, Equation 5.7 and Theorem 6. We generalise the notation for accumulated populations to accumulated products of populations: if $h(\boldsymbol{X}(t))$ is a product specification $h(\boldsymbol{X}(t)) = \prod_{i=1}^{o} X_i^{e_i}(t)$, we define the accumulation of the product as

$$\overline{h(\boldsymbol{X})}(t) = \int_0^t \prod_{i=1}^{o} X_i^{e_i}(u)\mathrm{d}u. \tag{5.11}$$

For example, we can write the above reward as $\mathcal{A}_{\text{network}}(t) = \overline{CS}(t)$.

We proceed analogously to the case of accumulated populations. We can swap differentiation and expectation in the ODE for an accumulated product of populations and get

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[\overline{h(\boldsymbol{X})}(t)] = \mathbb{E}[h(\boldsymbol{X}(t))]. \tag{5.12}$$

In our example, we get

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[\overline{CS}(t)] = \mathbb{E}[C(t)S(t)].$$

This ODE can be numerically solved together with the system for first- and second-order moments, giving us the mean of $\mathcal{A}_{\text{network}}(t)$.

In order to compute variance of $\mathcal{A}_{\text{network}}(t)$, we need to compute the moment $\mathbb{E}[\overline{CS}(t)^2]$. Similar to Equation 5.7, we can get an ODE for second-order moments of accumulated products of

populations:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[\overline{h_1(\boldsymbol{X})}(t)\overline{h_2(\boldsymbol{X})}(t)] = \mathbb{E}[h_1(\boldsymbol{X}(t))\overline{h_2(\boldsymbol{X})}(t)] + \mathbb{E}[h_2(\boldsymbol{X}(t))\overline{h_1(\boldsymbol{X})}(t)].$$

To compute the right-hand side terms, we need the following theorem

Theorem 7 (Generalisation of Theorem 6)**.** For finite products $h_i(\boldsymbol{X}(t))$, $i = 0, \ldots, n$ of the form

$$h_i(\boldsymbol{X}(t)) = \prod_{l=1}^{n} X_l(t)^{e_{i,l}} \qquad e_{i,l} \geq 0$$

defining a mixed moment specified by

$$h(\boldsymbol{X}(t)) = h_0(\boldsymbol{X}(t)) \prod_{j=1}^{n} \overline{h_j(\boldsymbol{X})}(t)^{e_j}$$

we have

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\left[h(\boldsymbol{X}(t))\right] = \mathbb{E}\left[f_{h_0}(\boldsymbol{X}(t)) \prod_{j=1}^{n} \overline{h_j(\boldsymbol{X})}(t)^{e_j}\right] + \sum_{k=1}^{n} \mathbb{E}\left[\frac{\partial h}{\partial \overline{h_k(\boldsymbol{X})}}(\boldsymbol{X}(t))\right]$$

where

$$\frac{\partial h}{\partial \overline{h_k(\boldsymbol{X})}}(\boldsymbol{X}(t)) = e_n \cdot h_0(\boldsymbol{X}(t)) \cdot h_k(\boldsymbol{X}(t)) \cdot \overline{h_k(\boldsymbol{X})}(t)^{e_k - 1} \prod_{j=1, j\neq k}^{n} \overline{h_j(\boldsymbol{X})}(t)^{e_j}$$

and $f_{h_0}(\cdot)$ is defined in Equation 3.3, that is a function such that

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[h_0(\boldsymbol{X}(t))] = \mathbb{E}[f_{h_0}(\boldsymbol{X}(t))].$$

Proof. See Appendix B.2 □

5.3.4 Completion times

We illustrate how to use moments of accumulated populations to obtain an approximation of completion time probabilities in PCTMC models. Consider a random variable representing the first time a reward $\mathcal{A}(t)$ hits a target value a:

$$T_c = \inf\{t \geq 0 : \mathcal{A}(t) \geq a\} \tag{5.13}$$

In order to guarantee various service level agreements of the form "the probability of reaching a reward a in time t is less than p", we are interested in the distribution of T_c, i.e. in the probabilities $\mathbb{P}(T_c \leq t)$. Section 3.5.4 described this problem for the case of completion times of populations – that is the time until a population reaches a target (usually specified by a proportion of the total number of agents). The technique uses the one sided improvement of Chebyshev's inequality: for

a random variable X:

$$\begin{aligned}\mathbb{P}(X - \mathbb{E}[X] \geq y) &\leq \frac{\text{Var}[X]}{\text{Var}[X] + y^2} \\ \mathbb{P}(\mathbb{E}[X] - X \geq y) &\leq \frac{\text{Var}[X]}{\text{Var}[X] + y^2}\end{aligned} \tag{5.14}$$

To get the required probabilities, we note that if $A(t)$ is non-decreasing (such as for example the reward $\mathcal{A}_{\text{energy}}(t)$):

$$\begin{aligned}\mathbb{P}(T_c \leq t) &= \mathbb{P}(\mathcal{A}(t) \geq a) \\ &= \mathbb{P}(\mathcal{A}(t) - \mathbb{E}[\mathcal{A}(t)] \geq a - \mathbb{E}[\mathcal{A}(t)])\end{aligned}$$

This allows us to use the following bounds:

$$\begin{aligned}\mathbb{P}(T_c \leq t) &\leq \frac{\text{Var}[\mathcal{A}(t)]}{\text{Var}[\mathcal{A}(t)] + (a - \mathbb{E}[\mathcal{A}(t)])^2} && \text{if } \mathbb{E}[\mathcal{A}(t)] \leq a \\ \mathbb{P}(T_c \leq t) &\geq 1 - \frac{\text{Var}[\mathcal{A}(t)]}{\text{Var}[\mathcal{A}(t)] + (a - \mathbb{E}[\mathcal{A}(t)])^2} && \text{if } \mathbb{E}[\mathcal{A}(t)] > a\end{aligned} \tag{5.15}$$

If $\mathcal{A}(t)$ cannot be guaranteed to be non-decreasing (such as the reward $\mathcal{A}_{\text{total}}(t)$), we have instead:

$$\mathbb{P}(T_c \leq t) \geq \mathbb{P}(\mathcal{A}(t) \geq a)$$

and only the lower bound in Equation 5.15 can be used. These bounds can be efficiently approximated by a solution to the systems of ODEs for accumulated populations, giving a lower and upper approximations to the CDF of completion times.

A more general result by Tari et al. [173] uses moments of order higher than two to produce tighter bounds on the CDF of rewards at each point in time, $\mathbb{P}(\mathcal{A}(t) \leq a)$. These bounds can replace the upper and lower bounds for respectively. We demonstrate the technique on a numerical example in Section 5.4.

5.3.5 Convergence of ODE approximations

Theorem 3 shows that a PCTMC $\boldsymbol{X}(t)$ converges to the deterministic solution of the respective mean ODEs as its initial populations get scaled to infinity. That is, we take a sequence of PCTMCs $\boldsymbol{X}^{(S)}(t)$ where each is constructed from $\boldsymbol{X}(t)$ by scaling its initial populations by a scale constant $S \in \mathbb{Z}_+$. Then as $S \to \infty$, each population mean re-scaled back by S, $\mathbb{E}[X_i^{(S)}(t)]/S$, converges to the ODE approximation $\widetilde{\mathbb{E}}[X_i(t)]$, uniformly for t in a finite interval $[0, t_f)$. Applying Fubini's theorem and dominated convergence, it is possible to extend this result to the convergence of accumulated populations. That is, as $S \to \infty$, the re-scaled means $\mathbb{E}[\overline{X_i^{(S)}}(t)]/S$ converge to the ODE approximations given by Equation 5.4. In the following chapter, Section 6.4 we formally show convergence for *hybrid PCTMCs*, which generalise the accumulated populations presented here.

In practice, this justifies an increased accuracy as the scale of the system increases, as further explored in Chapter 4. In Section 5.4, we proceed with similar investigations of moment ODEs for accumulated populations. We demonstrate the improved accuracy in mean approximations at larger scales. We also show that the second-order approximations from Section 5.3.2 are similarly accurate to population approximations. In particular, we show that the min-normal closure from Section 4.4 significantly improves the accuracy of mean and second-order moment approximations of accumulated populations.

5.3.6 Computational cost

The overall complexity of the presented technique to obtain moments of accumulated populations is given by the number of ODEs and the time over which they need to be numerically integrated. Calculating an n-th order moment of an accumulated population $\overline{X_i}(t)$ requires all the population moment ODEs of order up-to n. Additionally, this requires all the mixed ODEs of the form $h(\boldsymbol{X}(t))\overline{X_i}(t)^k$ where $k < n$ and $h(\boldsymbol{X}(t))$ is a population moment of order $n - k$. In total, the method gives $O(N^n)$ ODEs.

For example, in the client–server model, $N = 7$ (including the $think$ action-counting population) and to get the mean of $\mathcal{A}_{\text{total}}(t)$, the method requires 10 ODEs. For the variance of $\mathcal{A}_{\text{total}}(t)$, there are 65 ODEs – 7 for mean populations, 21 for second order population moments and 47 for mixed moments.

It is worth noting that the usual algorithms for numerically solving systems of ODEs, as described in Section 3.4.4, have both run time and memory requirements linearly dependent on the size of the system (unless stiffness problems arise). Therefore the technique is able to cope with fairly large systems. For example, models requiring more than 10^4 ODEs can be solved in under a minute on a standard Intel i7 3.0 GHz desktop computer.

5.4 Numerical examples

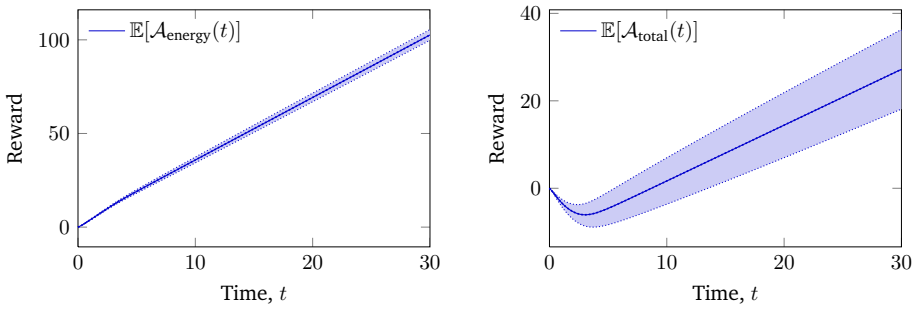
In this section we present numerical examples of ODE approximations of rewards in the client–server model. We use the same rate parameters as Model A in Table 4.1 with accumulation rate constants given in Table 5.1.

Table 5.1: Accumulation rate parameters in the client–server model. The rate r_{wakeup} is used in a further extension in Section 5.5.

r_S	r_{S_g}	r_{S_b}	c_{energy}	c_{fee}	r_{wakeup}
1.0	0.05	0.1	1.0	0.3	0.05

Figure 5.4 shows the evolution of means of the rewards $\mathcal{A}_{\text{energy}}(t)$ and $\mathcal{A}_{\text{total}}(t)$ over time. Additionally, the figure also shows an approximation to the 95% interval obtained from standard deviations of the rewards. The error of the ODE approximation when compared to estimates from 10^8 replications of stochastic simulation is very small relative to the scale of the figure.

Figure 5.5 investigates the error of the ODE approximations closely and looks at the accuracy of approximations of means and standard deviations at different scales of the system, $S = 1, 4, 16$



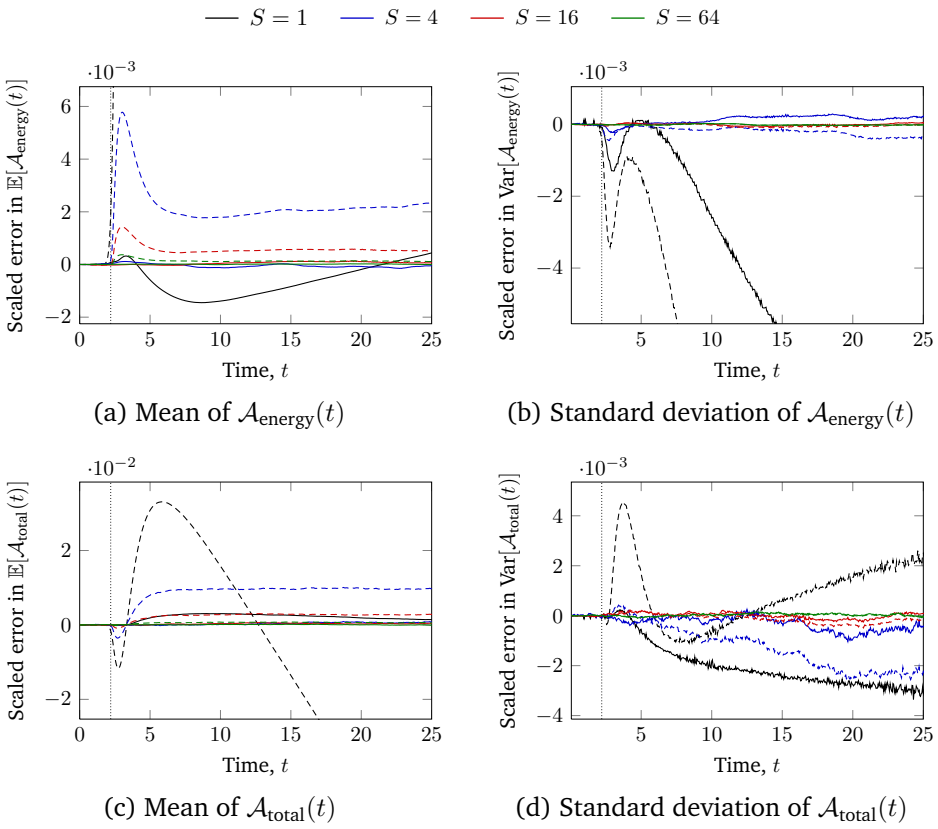
(a) Mean of $\mathcal{A}_{\text{energy}}(t)$ (b) Mean of $\mathcal{A}_{\text{total}}(t)$

Figure 5.4: Approximation of the rewards $\mathcal{A}_{\text{energy}}(t)$ and $\mathcal{A}_{\text{total}}(t)$. The shaded region shows an interval of width 1.95 of standard deviation of the respective reward, approximating the 95% interval. The dotted lines are estimates from 10^8 replications of stochastic simulation.

and 64. It compares the error of approximation when using two different closure schemes – the mean-field approximation from Section 3.4 and the min-closure from Section 4.4. As expected in Section 5.3.5, the scaled errors decrease as the system size increases. The error is affected by small switch-point distance (switch-point is shown by a vertical line). From this region, the error accumulates at a rate which is proportional to the error of the approximation of population means. In particular, we can see that the error stays at a constant level when the system moves away from a switch point. This corresponds to the accumulated error during the phase with small switch-point distance. Similar to the case of population moments, the min-closure significantly improves the accuracy of the ODE approximation (see Figure 4.13 for a similar comparison for population moments).

Figure 5.6 shows the lower and upper approximations to the CDF of the completion time of the reward $\mathcal{A}_{\text{energy}}(t)$ reaching a target value $a = 20$ for the first time. The bounds from second-order moments, as described in Section 5.3.4 can be used to estimate the completion time probabilities. For example, the probability of reaching the value a at time 6.0 is now between 0.95 and 1. On the other hand, the earliest time the reward reaches the value a with probability 0.95 is between 5.23 and 6.0. The bounds from the first seven moments computed using the method of Tari et al. [173] are much tighter – the probability of reaching a at time 6.0 is between 0.99 and 1.0 and the earliest time the reward reaches a with probability 0.95 is between 5.34 and 5.6. Figure 5.6b shows the error in the bounds approximation, as compared to the same bounds computed from moments from stochastic simulation. The maximum error when using the mean-field closure is in the order of 10^{-2} of the computed probability and the maximum error when using the min-closure is in the order of 10^{-3}.

The total reward $\mathcal{A}_{\text{total}}(t)$ can decrease over time and so only the lower bounds can be used to estimate the completion time probabilities. However, the upper bound is still a valid bound on the probability of the reward reaching a particular value at a given time (not necessarily for the first time). Figure 5.4 shows that the total reward initially goes negative. It would be useful to estimate the completion time when the reward hits a positive value for the first time. Figure 5.7 shows the approximate bounds on the lower bound of the probability of the reward $\mathcal{A}_{\text{total}}(t)$ reaching a small positive value 4.0 for the first time, computed using the first two moments in Chebyshev's inequality as well as using the first seven moments in the method of Tari et al. [173].



(a) Mean of $\mathcal{A}_{\text{energy}}(t)$ (b) Standard deviation of $\mathcal{A}_{\text{energy}}(t)$

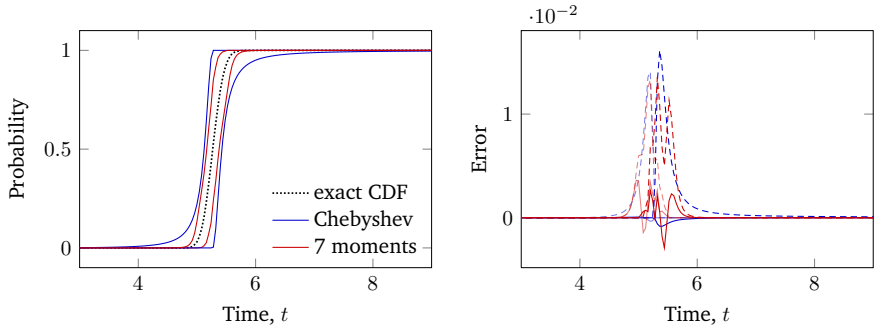
(c) Mean of $\mathcal{A}_{\text{total}}(t)$ (d) Standard deviation of $\mathcal{A}_{\text{total}}(t)$

Figure 5.5: Effects of scaling on the error of the ODE approximations of means and standard deviations of accumulated populations. The vertical line shows the location of a switch-point in the model, Section 4.2. The solid lines are approximations when using the min-closure from Section 4.4 and the dashed lines the approximations under the original mean-field closure, Section 3.4. Because of the high variance of rewards, 10^8 replications of stochastic simulation had to be used to obtain smoother plots.

The relatively higher variance of the total reward compared to the energy consumption reward causes the bounds to be less tight. For example, the probability of the reward becoming positive for the first time at time $t = 10$ is bounded by 0.06 from below whereas the actual probability is 0.59. The bound computed from the first seven moments only improve the Chebyshev bounds for time intervals around 3 units away from the time t when $\mathcal{A}_{\text{total}}(t)$ is zero. The error of the bounds is at the order of 10^{-3} as shown in Figure 5.7b.

5.5 Trade-off between energy consumption and performance

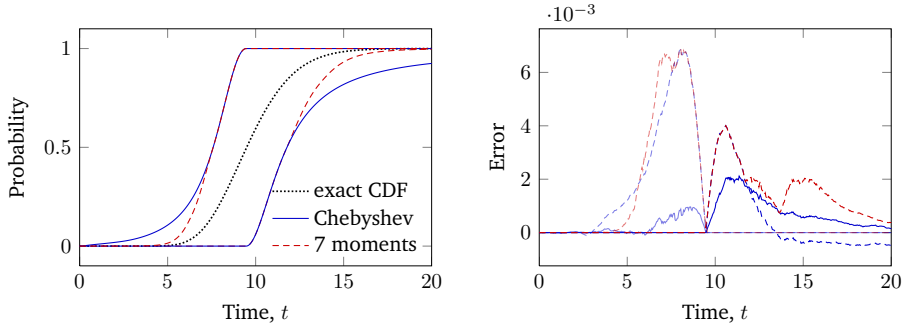
In practice, system providers cannot consider the client response times and server energy consumption in isolation. Usually, the performance required by clients is given in the form of a Service Level Agreement (SLA). This is often set as a constraint on a passage time probability. For example, in the client–server model, we could propose an SLA: “the probability of an individual client receiving the service within 13 seconds is at least 0.9”. An obvious target for the system is to be able to satisfy the SLA while at the same time operate as efficiently as possible. Usually, there is a trade-off between these two metrics. For example, increasing the number of servers leads to better response times (higher probability of finishing within a given time), but worse energy-efficiency.



(a) Bounds on completion time CDF

(b) Error in the bounds approximation

Figure 5.6: Approximations of the CDF of the time of the reward $\mathcal{A}_{\text{energy}}(t)$ reaching the target value $a = 2.0$. The dotted line is the exact CDF estimated from 10^8 replications of stochastic simulation. Figure (b) shows the absolute error, the difference between the bounds from simulation and ODE approximations. The dashed lines correspond to the error when using the mean-field closure and the solid lines when using the min-closure.



(a) Lower bound on the completion time CDF

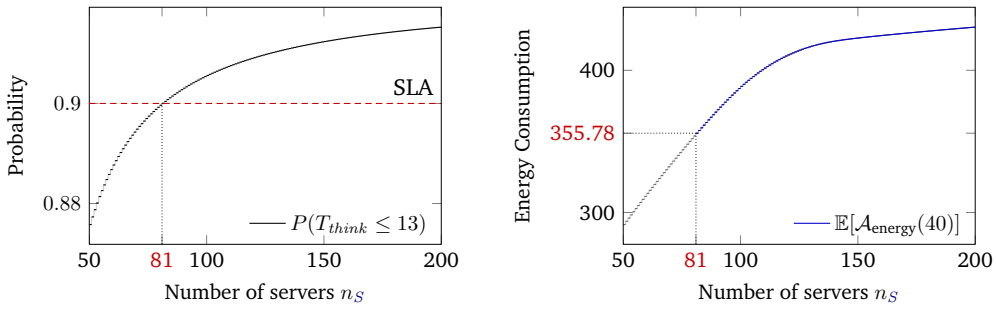
(b) Error in the lower bound

Figure 5.7: Approximations of the CDF of the time of the reward $\mathcal{A}_{\text{total}}(t)$ reaching the target value $a = 0.0$. The dotted line is the exact CDF estimated from 10^8 replications of stochastic simulation. Figure (b) shows the absolute error, the difference between the bounds from simulation and ODE approximations. The dashed lines correspond to the error when using the mean-field closure and the solid lines when using the min-closure.

One of the main benefits of our approach to the computation of reward metrics is that we are able to compute both passage time and reward metrics simultaneously. In particular, we are able to compute the passage time probabilities with the method [4] described in Section 3.5, while using the same set of ODEs to compute accumulated rewards. This allows us to efficiently capture the trade-off between performance (SLA satisfaction) and energy consumption (or in general minimisation/maximisation of a reward).

For example, Figure 5.8 considers the trade-off as the number of servers n_S in the client–server model increases from 50 to 200. For each number of servers, we compute the CDF of the time in which a client finishes a first *think* action, T_{think}, and plot the value at time $t = 13$, as given by the SLA, in Figure 5.8a. On the other hand Figure 5.8b shows the energy consumption at time $t = 40$, $\mathcal{A}_{\text{energy}}(40)$, computed from the same ODE solution. As expected, as the number of servers increases, both the probability of finishing but also the energy consumption increase. The SLA is satisfied if the probability of finishing on time is at least 0.9, as shown by the horizontal

line in Figure 5.8a. In that case, the optimal configuration is with $n_S = 81$ servers, achieving energy consumption of 355.78 units.



(a) Probability of finishing before time $t = 13.0$

(b) Energy consumption until time $t = 40.0$

Figure 5.8: Trade-off between response time and energy consumption as the number of servers increases.

5.5.1 Client–server model with server hibernation

When computing the trade-off in Figure 5.8, the relationship between energy consumption and the number of servers is obvious – it would not be necessary to compute the energy consumption reward in order to find the optimal number of servers, which clearly has to be minimal. However, in many cases the optimal configuration is not necessarily this intuitive. In fact, the reward we seek to optimise is not necessarily a monotonic function of the parameters that we can control. In this section, we illustrate this on a small extension of the client–server model, which allows the system to be more energy-efficient by hibernating servers according to the client demand.

One way to reduce energy consumption in the client–server model is to turn some of the servers off when the client demand decreases and turn them back on when the demand increases. We modify the GPEPA model from Section 3.2.3 accordingly. Idle servers can switch to a sleeping state at a rate independent of the number of requesting clients. This is indirectly related to the client demand, as these transitions have to compete with the $request$ synchronisations. Clients are additionally allowed to request a server to be switched back on with a synchronised action $wakeup$. See Figure 5.9 for the full GPEPA description of this modification. As before, we will be interested in the rewards $\mathcal{A}_{\text{energy}}(t)$ and $\mathcal{A}_{\text{total}}(t)$ and the time before a client finishes its first $think$ action. For simplicity, we assume that the energy consumption in the sleeping state is 0. We also ignore any additional energy costs of the switch from sleeping to idle state and vice versa. These could be easily added with additional impulse rewards.

We assume that the rate r_{sleep} and the number of servers n_S are the only parameters that the system providers can control – all the remaining rates and the number of clients depend on the infrastructure and the external client load. Our framework can be used to explore the energy–performance trade-off for a large number of combinations of the possible values of r_{sleep} and n_S. Figure 5.10 shows an example plot of the energy consumption and the total reward for parameter combinations where the SLA constraint is satisfied, as obtained by numerically solving the moment ODEs with initial values and rates computed from given parameters. In total, 1680

$$Client \stackrel{def}{=} (request, r_{request}).Client\_wait \\ +(wakeup, r_{wakeup}).Client$$

$$Server \stackrel{def}{=} (request, r_{request}).Server\_get \\ +(break, r_{break}).Server\_broken \\ +(sleep, r_{sleep}).Server_{sleep}$$

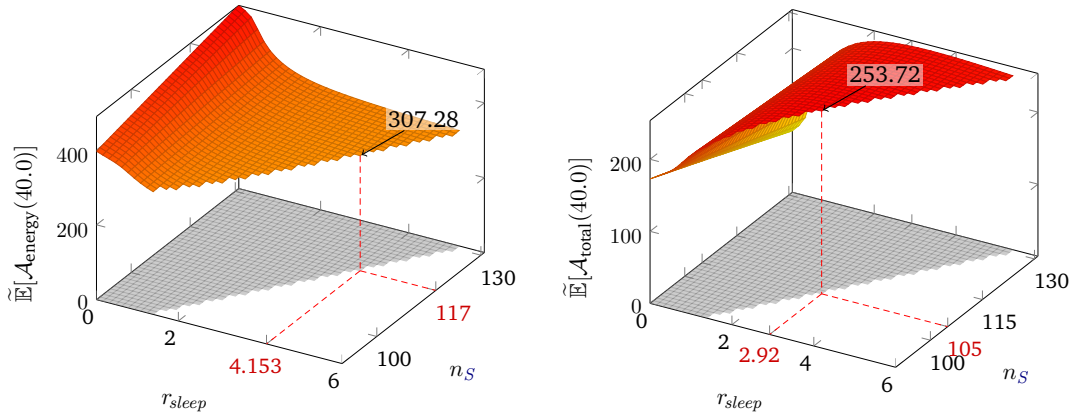
$$Client\_wait \stackrel{def}{=} (data, r_{data}).Client\_think$$
$$Client\_think \stackrel{def}{=} (think, r_{think}).Client$$

$$Server\_get \stackrel{def}{=} (data, r_{data}).Server$$
$$Server\_broken \stackrel{def}{=} (reset, r_{reset}).Server$$
$$Server_{sleep} \stackrel{def}{=} (wakeup, r_{wakeup}).Server$$

$$\mathbf{Clients}\{Client[n_C]\} \underset{\{request, data, wakeup\}}{\bowtie} \mathbf{Servers}\{Server[n_S]\}$$

Figure 5.9: Extension of the client–server GPEPA model allowing the servers to hibernate when the client demand is low. The modifications of the original model are highlighted.

different combinations are explored, with the overall computation time in the order of minutes on a standard desktop computer.



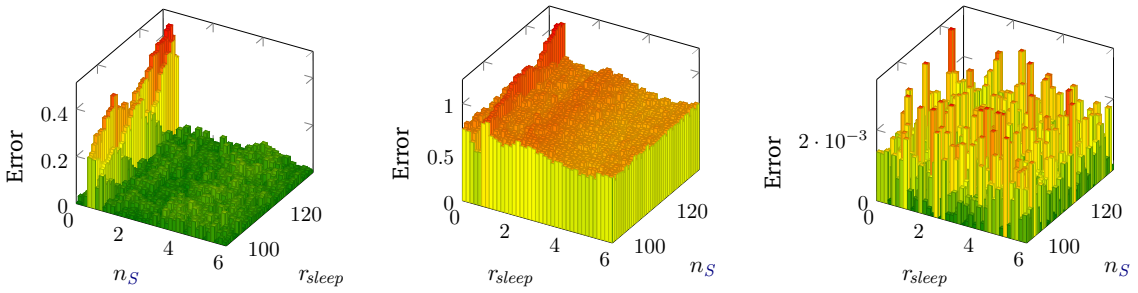
(a) Mean of $\mathcal{A}_{\text{energy}}(40)$ (b) Mean of $\mathcal{A}_{\text{total}}(40)$

Figure 5.10: Exploration of the trade-off between response time satisfaction and minimisation of energy consumption (a) / maximisation of the total reward (b) in the client–server model with server hibernation. The number of servers n_S is varied from 90 to 131 and the rate of sleeping r_{sleep} from 0.0 to 6.0 in 41 steps, giving a total of 1680 combinations. Only the points for configuration where the SLA given by $P(T_{think} \leq 13) \geq 0.9$ is satisfied are shown.

Intuitively, increasing the number of servers increases the probability of a client finishing early, but also raises the energy cost of running the system. Similarly, decreasing the hibernation rate has a positive effect on the system response. Figure 5.10a clearly shows that for each number of servers, there is a maximal hibernation rate under which the SLA is still satisfied. The minimum energy consumption lies on the boundary given by this relationship. From the numerical evaluations, the minimum energy consumption $\mathcal{A}_{\text{energy}}(40) = 307.28$ is achieved when the number of servers is $n_S = 117$ and the hibernation rate is $r_{sleep} = 4.15$. In case of the total reward $\mathcal{A}_{\text{total}}(t)$, there is an additional effect of the impulse reward from successful client requests. The maximum reward is $\mathcal{A}_{\text{total}}(40) = 253.72$ when $n_S = 105$ and $r_{sleep} = 2.92$.

Figure 5.11 shows the error of ODE approximation across all the considered parameter combinations, as compared to estimates from stochastic simulation. We note that simulation was feasible

only due to massive parallelisation provided by the GPA tool, which required around 100 CPU hours as opposed to several CPU minutes for the ODE analysis.



(a) $\mathbb{E}[\mathcal{A}_{\text{energy}}(40)]$ (b) $\mathbb{E}[\mathcal{A}_{\text{total}}(40)]$ (c) $P(T_{think} \leq 13)$

Figure 5.11: The error of approximations of accumulated rewards and passage times in Figure 5.10.

5.5.2 Global optimisation

Usually there are many more configuration parameters and it is not practical to explore the full parameter space such as in Figure 5.10. In general, the performance-energy trade-off in our setting can be expressed as a global optimisation problem:

Optimisation problem 1. Given a PCTMC $\boldsymbol{X}_{\boldsymbol{p}}(t)$ that depends on a vector of parameters $\boldsymbol{p} \in \boldsymbol{P} \subseteq \mathbb{R}^k$, we seek to optimise a reward that is a combination of accumulated rewards and population based rewards at a time t_f

$$\min_{\boldsymbol{p}\in\boldsymbol{P}} \mathbb{E}\left[\int_0^{t_f} h_1(\boldsymbol{X}_{\boldsymbol{p}}(u), \boldsymbol{p})\mathrm{d}u + h_2(\boldsymbol{X}_{\boldsymbol{p}}(t_f), \boldsymbol{p})\right]$$

where $h_i(\boldsymbol{X}(t), \boldsymbol{p})$ are linear combinations of $X_j(t)$. Initial populations are given by $\boldsymbol{X}_{\boldsymbol{p}}(0) = \boldsymbol{Y}_{\boldsymbol{p}}$. Constraints are imposed by a set of SLA inequalities on points on CDF of passage times T_i

$$P(T_i \leq t_i) \geq c_i \qquad \text{for } 1 \leq i \leq m$$

Using the ODE approximation from this chapter, we can turn this into a global optimisation problem with an *embedded system of ODEs*.

Optimisation problem 2. Let $\boldsymbol{x}_{\boldsymbol{p}}(t)$ be the ODEs for moments of populations and $\boldsymbol{y}_{\boldsymbol{p}}(t)$ moments of accumulated populations and mixed moments in a PCTMC $\boldsymbol{X}_{\boldsymbol{P}}(t)$, that is

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}_{\boldsymbol{p}}(t) = \boldsymbol{f}_1(\boldsymbol{x}_{\boldsymbol{p}}(t), \boldsymbol{p})$$
$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{y}_{\boldsymbol{p}}(t) = \boldsymbol{f}_1(\boldsymbol{x}_{\boldsymbol{p}}(t), \boldsymbol{y}_{\boldsymbol{p}}(t), \boldsymbol{p})$$

where functions $\boldsymbol{f}_i$ are determined by the ODE approximation. The initial values are $\boldsymbol{x}_{\boldsymbol{p}}(0) = \boldsymbol{m}(\boldsymbol{Y}_{\boldsymbol{p}})$ and $\boldsymbol{y}_{\boldsymbol{p}}(0) = 0$ where $\boldsymbol{m}(\cdot)$ is a function which initialises the moments from the initial values of populations. We seek to optimise

$$\min_{\boldsymbol{p}\in\boldsymbol{P}} h_1(\boldsymbol{y}_{\boldsymbol{p}}(t_f), \boldsymbol{p}) + h_2(\boldsymbol{x}_{\boldsymbol{p}}(t_f), \boldsymbol{p})$$

under constraints

$$g_i(\boldsymbol{x_p}(t_i)) \geq c_i \qquad \text{for } 1 \leq i \leq m$$

where the functions $g_i(\cdot)$ approximate the CDFs of passage times T_i.

Unfortunately, the structure of this optimisation problem is too general for most existing efficient solution techniques. The objective function is not guaranteed to be convex. There are existing techniques for solving global optimisation problems with embedded ODEs, such as the algorithm of Singer and Barton [165]. However, the right-hand sides of the moment equations can be complicated and contain polynomial functions and occurrences of the $\min$ function and normal PDF and CDF, preventing a direct application of the algorithm. Another problem is the presence of integer parameters such as the initial populations in the client–server example. Nevertheless, the ODE analysis allows us to efficiently compute individual instances of the objective function and constraints and therefore make it possible to apply approximate numerical global optimisation algorithms. Particularly suitable are derivative-free methods [161] which only require the ability to evaluate the objective function and not its derivatives. We have chosen the *Constrained Optimisation By Linear Approximation (COBYLA2)* algorithm by Powell [154] due to its general applicability and available implementation and integrated it into the GPA tool. For example, running the algorithm on the example from Figure 5.10 when allowing real valued initial populations, requires only 76 evaluations of the objective function and constraints, arriving at an approximate minimal energy consumption of 307.1 with parameters $n_S = 116.4$ and $r_{sleep} = 4.13$. If we instead iterate over the possible integer valued initial populations and run the optimisation algorithm at each step, we obtain an improved minimal consumption 307.09 with $n_S = 116$ and $r_{sleep} = 4.09$ in around 600 evaluations of the objective function.

5.6 Estimating power consumption rates

The power consumption rates we used in this chapter such as r_S, r_{S_g}, r_{S_b} in the client–server example were chosen only for illustration purposes. Techniques for creating accurate power models of real systems form a wide research area and lie outside of the scope of this thesis. In this section, we review a number of possibilities that could be used within the PCTMC framework. From a high-level perspective, power models can be obtained by a combination of direct measurements, indirect estimation from various metrics such as hardware performance counters and methods to extrapolate models into larger systems or different application domains. Mobius et al. [147] give an overview of power estimation models for processors and servers. The selection of available techniques depends on the context in which the PCTMC framework is used. In Section 9.2 we suggest how PCTMC models could be used at various stages of the system life cycle. At a purely design stage, PCTMC models could be used to evaluate possible configurations of the system. In such case, it is often impossible to perform measurement experiments on the desired hardware and software combinations. Estimates of power consumption rates can be obtained from hardware specification. For example, Intel includes the *thermal design power* in their CPU specification [112]. This can be used to obtain a crude estimate of the upper

bound of the power consumption of a CPU, which usually forms a significant part of the power consumption of the whole system [147]. At this stage, the remaining parameters in the model (such as job arrival rates) are also rather speculative and the analysis results only suitable for relative comparison of different configurations.

In some cases, the modeller has access to some of the hardware already at the design stage of the system. For example, in a large data centre, there are often only a few different types of servers. It is possible to use a small number of each type of machine to obtain power consumption benchmarks that could be used in the model of the whole system [133].

Section 9.2 also suggests a dynamic usage of PCTMC models that is evaluated alongside a live system and can be used for accurate predictions of the system behaviour in the immediate future. It would be possible to instrument the system (or a small subset of the system) to provide live measurement of power consumption metrics. These could be used to accurately calibrate the power consumption parameters. For example, *Joulemeter* [115] is a software for dynamically learning an accurate model of power consumption of a laptop or desktop computer.

In all the mentioned examples, the accuracy of the resulting power consumption estimate depends on the detail of the model. Even if low level details of the system such as CPU cache memory and network card are captured, the overall error of a power model is highly workload dependent [143]. It is part of the modelling challenge to choose a model structure which accurately captures the power consumption behaviour (as well as other operational aspects of the real system) while maintaining a manageable size of the resulting PCTMC.

5.7 Conclusion

In this chapter, we have shown how to extend the efficient ODE analysis of PCTMCs to additionally capture accumulated rewards. These model important operational metrics of systems such as total energy consumption or generated income. Impulse accumulated rewards can be analysed as populations after a simple transformation of the PCTMC. We have derived additional differential equations for means and higher-order moments of continuously accumulated integrals of populations. The latter equations require auxiliary ODEs capturing mixed moments that are expectations of products of populations and accumulated populations. These can be also used when computing moments of rewards which are combinations of continuously accumulated rewards and impulse rewards. We have also shown how to use the higher-order moments of rewards to approximate bounds on distributions of completion times until a reward reaches a target value. We have demonstrated the techniques on a reward structure defined on the client–server model from Section 3.2.3. We have numerically investigated the accuracy of the ODE approximation in a style similar to Chapter 4, comparing different moment closure schemes and looking at different scales of the system.

A crucial advantage of our approach is that we are now in a position to compute reward and response time metrics during the same analysis. This allows us to accurately capture trade-offs such as those between energy consumption and performance, given by the response time probabilities of the system that can be already computed within the framework. We

have formulated a global optimisation problem where the objective function is given as an accumulated reward, such as energy in a server farm, and constraints are given by lower bounds on passage time probabilities, such as service level guarantees by the service provider. Using the moment ODE equations, this problem can be approximated by a global optimisation problem with embedded differential equations. While such a problem is still generally not analytically tractable, the efficient evaluation of the objective function and constraints allows us to use approximate numerical algorithms. We have demonstrated on the client–server reward model how an application of such algorithms can efficiently find an approximate solution to the optimisation problem.

We use the results from this chapter in two ways in the subsequent chapters of this thesis. In Chapter 6, we further extend this framework to allow accumulations of a more general class of expressions as opposed to linear combinations of populations. We show how to model feedback from rewards by allowing the transition rates to contain rewards. In Chapter 7, we define an extension of GPEPA process algebra which more accurately captures ongoing transactions between different agents. These leads to PCTMCs with a large number of populations and a complicated structure of transition rates. Our efficient implementation of the reward ODE analysis within the GPA tool, described in Chapter 8 makes it possible to address the performance-energy trade-off in such large models.

Chapter 6

Hybrid PCTMCs

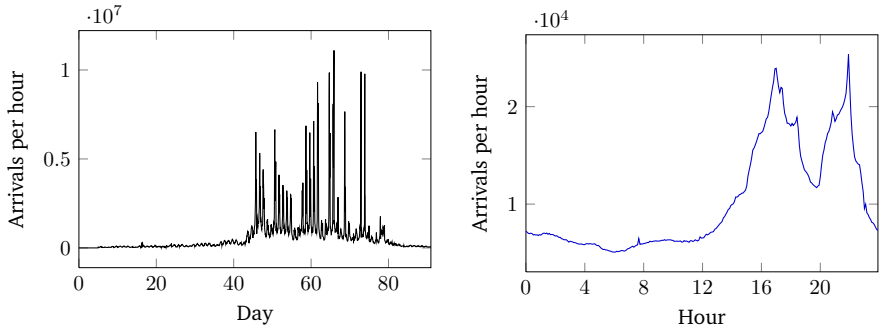
Key contributions		
*h*PCTMC – PCTMC with continuous variables	6.2	[15, 14]
Extended ODE analysis to moments in *h*PCTMC	6.3.1	[15, 14]
*h*PCTMC Time-inhomogeneous models and dynamic SLA verification	6.6	[14]

6.1 Introduction

The behaviour of large computing clusters is often indirectly controlled by feedback from various accumulated continuous quantities, such as temperature, energy consumption or the total operational cost. For example, an air conditioning controller in a server farm will react to the ambient temperature. At the same time, sophisticated thermally-aware schedulers [172] can use temperature sensors to regulate server operation and thus indirectly affect the ambient temperature, creating a feedback loop. Additionally, scheduling can be affected by time-varying workloads. These might be driven by seasonal job submission patterns or the need to follow the fluctuating price of electricity or availability of renewable energy sources [135].

In this chapter, we address these issues within the PCTMC framework and provide efficient techniques to analyse models with feedback from continuous variables and with time-depended rates. This is an important step towards applying the PCTMC framework and the associated ODE analyses to real data. For example, Figure 6.1a shows the load experienced by the World Cup 1998 website [137]. Our framework can be used to verify whether a complex large-scale model of a computing cluster serving the website would cope with such a workload, while satisfying pre-determined service level agreements (SLA) with its users. The SLAs often guarantee maximal execution time for a task with some agreed probability. These are usually assumed to hold in the steady state of the system and can be efficiently verified within the ODE analysis framework [4]. If parameters of the system are changing over a period of time, we need to check the validity of SLAs throughout this period. We present dynamic SLA verification and show how to use the efficient ODE analysis to check passage-time probabilities for clients arriving into the system at a number of pre-determined time points. At the same time, the framework is able to incorporate feedback from continuous quantities. The ODE analysis can compute the total energy consumption of the servers as in Chapter 5 and also take into account the cost of running air conditioning units that control the operating temperature of the system.

The main contributions of this chapter are:



(a) Accesses each day (b) Accesses on an average day

Figure 6.1: The number of accesses of the World Cup 1998 website during a long period around the event (a) and hourly accesses on an average day (b) during the tournament.

***h*PCTMC – Hybrid PCTMC** In Section 6.2 we define an extension of PCTMC with state space that includes a vector of continuous variables, obtaining a hybrid model as described in Section 2.4. The continuous variables generalise the accumulated populations in Chapter 5 and can instead grow as a function of populations and also influence populations by appearing in the rates of transition classes in a PCTMC. The continuous variables are global and their value is available to every agent, as opposed to agents having individual continuous variables [e.g. 51, 97]. We demonstrate how to use *h*PCTMC to model systems with *feedback* from continuous variables. In Section 6.3.1 we extend the ODE analysis from Section 3.4 to *h*PCTMC and show how to derive approximations to moments of populations and continuous variables. Section 6.4 justifies this approach by proving convergence to the solution of the mean-field equations as the scale of the system increases. We show that the min-normal moment closure from Section 4.4 is particularly suited to models with threshold feedback from continuous quantities and leads to increased accuracy of the ODE analysis. Section 6.5 presents a worked example of a heterogeneous computing cluster with controlled temperature in Section 6.5. We validate a large number of system evaluations against stochastic simulation.

Time-inhomogeneous rates In Section 6.6 we show how to incorporate time-inhomogeneous rates in *h*PCTMC models. These are crucial when applying the framework to real-world examples, where workloads often depend on time. We show how to combine passage-time techniques of Hayden et al. [4] with time-inhomogeneous rates to dynamically verify SLAs at a number different points throughout the evolution of the system. Section 6.6.2 shows a worked example with time-inhomogeneous rates from real data and applies the dynamic SLA verification to find a suitable system size that minimises energy consumption while maintaining a required performance.

We have extended the implementation in the GPA tool, Chapter 8, with developments in this chapter. The tool implements the *h*PCTMC framework and allows inclusion of data-driven time-inhomogeneous rates.

6.2 Hybrid PCTMCs

In this section, we define an extension of a continuous-time Markov population process that treats accumulated quantities such as the rewards in Chapter 5 as part of the state space of the model.

> We illustrate the following definitions on a PCTMC representing a simplified, single-stage version of the client–server model from Section 3.2.3. The GPEPA description of the model is
>
> $$\begin{array}{lcllcl} Client_0 & \stackrel{def}{=} & (data, r_{data}).Client_1 & Server_0 & \stackrel{def}{=} & (data, r_{data}).Server_1 \\ Client_1 & \stackrel{def}{=} & (task, r_{task}).Client_0 & Server_1 & \stackrel{def}{=} & (reset, r_{reset}).Server_0 \end{array}$$
>
> $$\mathbf{Clients}\{Client_0[n_C]\} \underset{data}{\bowtie} \mathbf{Servers}\{Server_0[n_S]\}$$
>
> The state space of the underlying PCTMC consists of numerical vectors $\boldsymbol{X} = (C_0, C_1, S_0, S_1) \in \mathbb{Z}_+^4$, where the populations correspond to the four possible group–agent pairs, and the initial state given by the system equation is $\boldsymbol{X_0} = (n_C, 0, n_S, 0)$. There are three transition classes in this model – one corresponding to the synchronised event where a client sends its data to a server and two independent events where a client or a server move to their initial states. According to PCTMC semantics of GPEPA, Section 3.2.2, the respective difference vectors and rate functions are $\boldsymbol{\delta}_1 = (-1, -1, 1, 1)$ with $r_1(\boldsymbol{X}) = \min(C_0, S_0) r_{data}$, $\boldsymbol{\delta}_2 = (1, -1, 0, 0)$ with $r_2(\boldsymbol{X}) = C_1 \cdot r_{task}$ and $\boldsymbol{\delta}_3 = (0, 0, 1, -1)$ with $r_3(\boldsymbol{X}) = S_1 \cdot r_{reset}$.

We augment the state space with a set of continuous variables governed by an auxiliary system of integral equations whose evolution may additionally depend on the discrete populations. The continuous variables can be used to track the evolution of associated quantities such as energy consumption or ambient temperature. The rates of the Markovian evolution of the discrete populations may also depend on the value of these variables, thus allowing, for example, the temperature to enter a feedback loop controlling the system.

6.2.1 Definition

The state space of a *hybrid PCTMC* (*hPCTMC*) is a subset of $\mathbb{Z}_+^N \times \mathbb{R}^M$ consisting of states $(\boldsymbol{X}, \boldsymbol{\mathcal{Y}})$, where $\boldsymbol{X} \in \mathbb{Z}_+^N$ captures discrete populations and $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^M$ captures continuous variables. The discrete populations evolve as in traditional PCTMCs, Section 3.1, that is according to a set $\mathcal{C}$ of transition classes. The associated rate functions are extended onto the full state space, that is $r_c \colon \mathbb{Z}_+^N \times \mathbb{R}^M \to \mathbb{R}_+$. The initial state of $\boldsymbol{X}$ and $\boldsymbol{\mathcal{Y}}$ is given by the (possibly correlated) random variables $\boldsymbol{X_0}$ and $\boldsymbol{\mathcal{Y}_0}$.

The evolution of continuous variables $\boldsymbol{\mathcal{Y}}(t)$ is given by an integral equation of the form:

$$\boldsymbol{\mathcal{Y}}(t) = \boldsymbol{\mathcal{Y}}_0 + \int_0^t \boldsymbol{g}(\boldsymbol{X}(s), \boldsymbol{\mathcal{Y}}(s))\, \mathrm{d}s \tag{6.1}$$

where $\boldsymbol{g} : \mathbb{Z}_+^N \times \mathbb{R}^M \to \mathbb{R}^M$ is an accumulation function. That is, the continuous variables $\boldsymbol{\mathcal{Y}}(t)$ are deterministically accumulated between successive jumps in the population process $\boldsymbol{X}(t)$.

In the client–server model we might wish to model generation of heat by servers in the active state $Server_1$, resulting in an increase in the total energy in the server room. In order to model the heating–cooling process, we extend the discrete model with a temperature variable $\mathcal{T}$ defined below. We add a group of air conditioning units which control the temperature:

$$Aircon_0 \stackrel{def}{=} (on, \lambda_{on}(\mathcal{T})).Aircon_1 \qquad Aircon_1 \stackrel{def}{=} (off, \lambda_{off}(\mathcal{T})).Aircon_0$$

$$\Big(\mathbf{Clients}\{Client_0[n_C]\} \underset{data}{\bowtie} \mathbf{Servers}\{Server_0[n_S]\}\Big) \parallel \mathbf{Aircon}\{Aircon_0[n_A]\}$$

where the rates λ_{on} and λ_{off} are defined below. The active air conditioning units contribute to the cooling of the environment, by transferring heat out of the room. If we assume that the heat generation and cooling rates for a single server and air conditioning unit (r_{heat} and r_{cool}) are constant over time, the thermal energy in the server room can be captured by an accumulated variable $\mathcal{E}$:

$$\mathcal{E}(t) = \mathcal{E}_0 + \int_0^t r_{heat} S_1(u) - r_{cool} A_1(u)\,\mathrm{d}u$$

where $\mathcal{E}_0$ is the initial energy in the room.

We can introduce feedback into the system by making the air conditioning transition rates depend on the current temperature of the room. An approximate physical model for the temperature is:

$$\mathcal{T}(t) = \frac{c}{v}\mathcal{E}(t) \tag{6.2}$$

where c is a constant and v is the total volume of air in the room. One possible control policy for the air conditioning units might be: when the temperature is above a given threshold t_{thresh}, units switch on at a rate proportional to the difference between the temperature and the threshold, otherwise active units switch off after an exponentially distributed time period:

$$\begin{aligned} \lambda_{on}(\mathcal{T}) &= r_{on}(\mathcal{T} - t_{thresh})^+ \\ \lambda_{off}(\mathcal{T}) &= r_{off} \end{aligned} \tag{6.3}$$

6.2.2 Regularity conditions

In general, each *h*PCTMC process can be realised as a *piecewise deterministic Markov process (PDMP)* [64]. However, in order for the above construction to result in a uniquely well-defined PDMP on any finite interval of time, some regularity conditions are required. In particular, it is important that the possibility of infinitely many jumps of the discrete component in a finite period of time is prevented and also that the continuous component cannot grow unboundedly in a finite period of time, that is it cannot explode. The following conditions are sufficient to achieve this, where $\mathbb{X} \subset \mathbb{Z}_+^N$ is defined to be the reachable state space, in a finite time horizon t_f, of the discrete part of the state space:

1. There exist constants $A, B \in \mathbb{R}_+$ such that for all $\boldsymbol{x} \in \mathbb{X}$, $\boldsymbol{y} \in \mathbb{R}^M$ and $c \in \mathcal{C}$:

$$\|\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y})\| \leq A(\|\boldsymbol{y}\| + 1) \quad \text{and} \quad r_c(\boldsymbol{x}, \boldsymbol{y}) \leq B(\|\boldsymbol{y}\| + 1)$$

2. For a fixed population vector $\boldsymbol{x} \in \mathbb{X}$, the function $\boldsymbol{g}(\boldsymbol{x}, \cdot)\colon \mathbb{R}^M \to \mathbb{R}^M$ satisfies a local Lipschitz condition;

3. For each transition class $c \in \mathcal{C}$, the function $r_c(\boldsymbol{x}, \cdot)\colon \mathbb{R}^M \to \mathbb{R}_+$ is measurable for each $\boldsymbol{x} \in \mathbb{X}$.

Assumptions 1 and 2 guarantee that, between successive discrete jumps, the continuous component is defined uniquely as the absolutely continuous solution to Equation 6.1 which exists as long as it does not explode [e.g. 60, Chapter 2]. In fact, the only way that the above construction can fail is if the continuous component explodes, since, otherwise, the maximal jump rate is bounded by assumption 1. However, if the continuous component does explode, say, at time t^*, then for any $t < t^*$, we have:

$$\|\boldsymbol{\mathcal{Y}}(t)\| \leq \|\boldsymbol{\mathcal{Y}}_{\mathbf{0}}\| + \int_0^t \|\boldsymbol{g}(\boldsymbol{X}(s), \boldsymbol{\mathcal{Y}}(s))\| \,\mathrm{d}s \leq \|\boldsymbol{\mathcal{Y}}_{\mathbf{0}}\| + At^* + \int_0^t A\|\boldsymbol{\mathcal{Y}}(s))\| \,\mathrm{d}s$$

Applying a version of Grönwall's lemma [e.g. 72, Page 498] yields:

$$\|\boldsymbol{\mathcal{Y}}(t)\| \leq \, (\|\boldsymbol{\mathcal{Y}}_{\mathbf{0}}\| + At^*) \exp(At)$$

This implies that $\boldsymbol{\mathcal{Y}}(t)$ cannot explode at time t^* since it is continuous and bounded by $(\|\boldsymbol{\mathcal{Y}}_{\mathbf{0}}\| + At^*) \exp(At^*)$ for any $t < t^*$. Thus we have a contradiction and have shown that, subject to the assumptions above, our construction is well-defined on finite intervals of time.

It can be seen that the client–server model satisfies these requirements. In Section 6.6 we show how a continuous quantity can be used to describe time in the system and thus allow the model to include time-inhomogeneous behaviour. In particular, in models in Section 6.6 and Section 6.6.2, we will use time-varying parameters obtained as piecewise continuous functions from measured data. These will have only a finite number of jumps and so all the conditions hold in such models.

6.3 ODE analysis

It is possible to extend the simulation algorithm for PCTMCs, for example described in Section 3.3, to realise traces of the discrete and continuous state components of *h*PCTMC models. Despite efficient algorithms for exact simulation of such models, e.g. [18], the simulation still suffers from high computational costs in case of large systems. In order to produce numerical examples in this chapter, we have implemented an extension of the stochastic simulation algorithm from Section 3.3 which deals with rates that can vary over the time between two successive events (required when a transition rate refers to a continuous variable).

6.3.1 Mean-field approximations

We extend the efficient mean-field analysis of PCTMC models [e.g. 99, 108, 183], described in Section 3.4 to the case of *h*PCTMC models. Specifically, we define a function $\boldsymbol{f} : \mathbb{R}^N \times \mathbb{R}^M \to \mathbb{R}^N$, analogous to Equation 3.7, as

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}) := \sum_{c \in \mathcal{C}} r_c(\boldsymbol{x}, \boldsymbol{y}) \boldsymbol{\delta}_c$$

for suitable real extensions of the rate functions r_c and also implicitly consider a real extension of the accumulation function $\boldsymbol{g}$. Then an extension of the mean-field approach yields the following system of $M + N$ differential equations for $\boldsymbol{x}(t) \in \mathbb{R}^N$ and $\boldsymbol{y}(t) \in \mathbb{R}^M$:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t) &= \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{y}(t)) \\ \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{y}(t) &= \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{y}(t)) \end{aligned} \tag{6.4}$$

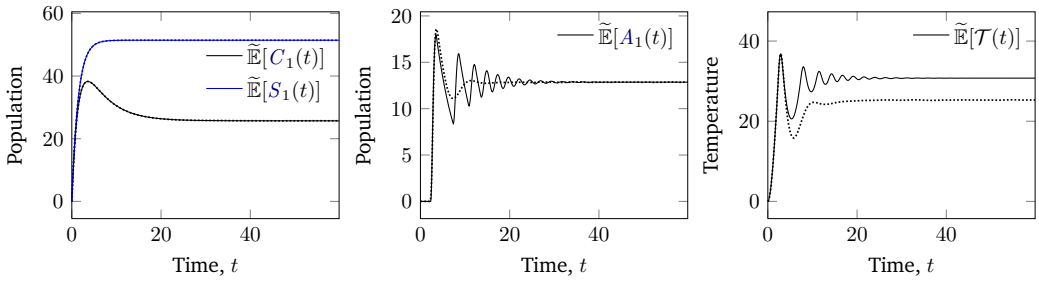
with initial conditions $\boldsymbol{x}(0) = \mathbb{E}[\boldsymbol{X_0}]$ and $\boldsymbol{y}(0) = \mathbb{E}[\boldsymbol{Y_0}]$. The solution of this system can be interpreted as an approximation of means of the stochastic processes $\boldsymbol{X}(t)$ and $\boldsymbol{\mathcal{Y}}(t)$, respectively, or for sufficiently large populations, as an approximation to individual traces of the stochastic processes. In Section 6.4 we show that, in the limit of large populations, the traces of the processes $\boldsymbol{X}(t)$ and $\boldsymbol{\mathcal{Y}}(t)$ (and in particular the means $\mathbb{E}[\boldsymbol{X}(t)]$ and $\mathbb{E}[\boldsymbol{\mathcal{Y}}(t)]$) converge to the mean-field solutions $\boldsymbol{x}(t)$ and $\boldsymbol{y}(t)$, respectively. We will extend the notation from Section 3.4 and write $\widetilde{\mathbb{E}}[\boldsymbol{\mathcal{Y}}(t)]$ for $\boldsymbol{y}(t)$.

> In the client–server model, the mean-field system includes equations such as:
>
> $$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[S_0(t)] &= r_{reset}\widetilde{\mathbb{E}}[S_1(t)] - r_{data}\min(\widetilde{\mathbb{E}}[C_0(t)], \widetilde{\mathbb{E}}[S_0(t)]) \\ \frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[\mathcal{E}(t)] &= r_{heat}\widetilde{\mathbb{E}}[S_1(t)] - r_{cool}\widetilde{\mathbb{E}}[A_1(t)] \\ \frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[Aircon_1(t)] &= (\widetilde{\mathbb{E}}[\mathcal{T}(t)] - t_{thresh})^+ \cdot r_{on}\widetilde{\mathbb{E}}[Aircon_0] - r_{off}\widetilde{\mathbb{E}}[Aircon_1(t)] \end{aligned}$$

Figure 6.2 shows the numerical solutions to the mean-field ODEs from Equation 6.4 as applied to the client–server model, compared to estimates of the exact means sampled from 10^5 simulation runs. As in the rest of this thesis, the estimates from simulation are shown as dotted lines unless noted otherwise. Table 6.1 shows the specific values of parameters used to produce this figure and all the subsequent examples from this model.

Table 6.1: Rate and threshold parameters used in the client–server model with air conditioning units.

r_{data}	r_{task}	r_{reset}	r_{on}	r_{off}	r_{heat}	r_{cool}	t_{thresh}
0.6	0.2	0.1	0.2	0.2	0.2	0.4	30



(a) Active clients/servers (b) Active air conditioning units (c) Temperature

Figure 6.2: Approximation of means of populations and of the temperature variable in the client–server model with air conditioning units.

6.3.2 Higher-order moments

We proceed with extending the ODE analysis of higher moments of populations in PCTMCs [e.g. 99, 10, 81], as described in Section 3.4.2, to higher moments of populations and continuous variables in *h*PCTMCs. The following theorem is a *h*PCTMC version of Theorem 1:

Theorem 8. Let $(\boldsymbol{X}(t), \boldsymbol{\mathcal{Y}}(t), \mathcal{C}, \boldsymbol{g}, \boldsymbol{X_0}, \boldsymbol{\mathcal{Y}_0})$ be a *h*PCTMC and $h : \mathbb{R}^N \times \mathbb{R}^M \to \mathbb{R}$ a continuous bounded function differentiable in the latter M variables. The expectation of $h(\boldsymbol{X}(t), \boldsymbol{\mathcal{Y}}(t))$ can be described by a differential equation

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[h(\boldsymbol{X}(t), \boldsymbol{Y}(t))] = \sum_{i=1}^{M} \mathbb{E}\left[g_i(\boldsymbol{X}(t), \boldsymbol{\mathcal{Y}}(t)) \frac{\partial h}{\partial \mathcal{Y}_i(t)}(\boldsymbol{X}(t), \boldsymbol{\mathcal{Y}}(t))\right] + \sum_{c \in \mathcal{C}} \mathbb{E}\left[r_c(\boldsymbol{X}(t), \boldsymbol{\mathcal{Y}}(t)) \left(h(\boldsymbol{X}(t) + \boldsymbol{\delta}_c, \boldsymbol{\mathcal{Y}}(t)) - h(\boldsymbol{X}(t), \boldsymbol{\mathcal{Y}}(t))\right)\right] \quad (6.5)$$

with initial value $\mathbb{E}[h(\boldsymbol{X_0}, \boldsymbol{\mathcal{Y}_0})]$.

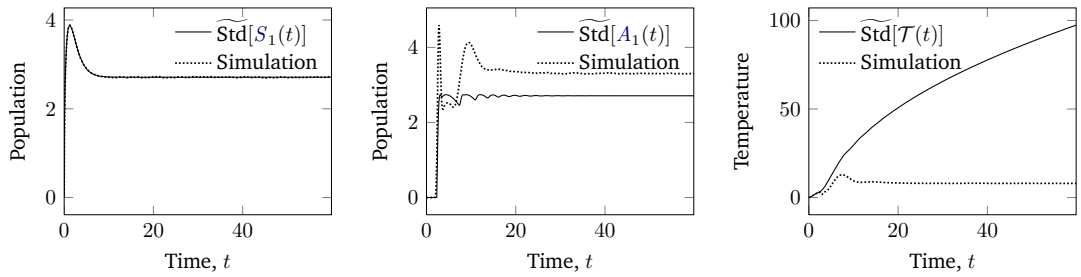
Proof. See Appendix C.1 □

Equations for second-order moments can be obtained by choosing $h(\boldsymbol{x}, \boldsymbol{y}) := x_i y_j$, $x_i x_j$ and $y_i y_j$ for each appropriate i and j. Assuming that the set of possible population vectors $\mathbb{X}$ is finite then the arguments of Section 6.2 guarantee that, over finite intervals of time, the process $(\boldsymbol{X}(t), \boldsymbol{\mathcal{Y}}(t))$ is bounded to remain in some compact set, and then the boundedness requirement for the functions h need only be honoured on this set. Monomial functions of any order can be used to obtain equations for arbitrary order moments.

If the functions $\boldsymbol{f}$ and $\boldsymbol{g}$ are non-linear, as is usually the case in GPEPA models for example, the terms on the right-hand side of Equation 6.5 involve expectations of non-linear functions of populations and accumulations and thus needs to be simplified by applying some form of moment-closure approximation, in the same fashion as in Section 3.4.2.

> In the client–server model, the right-hand side of Equation 6.5 contains terms of the form $\mathbb{E}[\min(C_0(s), S_0(s))]$. In Equation 6.4 above, the approximation from Equation 3.5 has been used, giving $\min(\mathbb{E}[C_0(s)], \mathbb{E}[S_0(s)])$. This has been shown in Section 4.2 to work well in

general for a large class of performance models. However, if the process remains close to states where the arguments of the minimum function are equal, that is the so-called *switch point distance* is small for a long period of time, the accuracy of this approximation can decrease significantly. This can be crucial in *h*PCTMC models where the minimum function is used in rates which control the continuous variables. In the client–server model, the control rate $(\mathcal{T}(s) - t_{thresh})^+$ can be expressed as $-\min(-\mathcal{T}(s) + t_{thresh}, 0)$. The behaviour of the air conditioning agents results in the temperature variable staying near the threshold value and thus in small switch-point distance. The resulting error is visible in the approximation of the mean of the temperature variable, as shown in Figure 6.2c. In Section 6.4.3, we show how the min-normal closure from Section 4.4 improves this approximation.



(a) Active servers (b) Active aircon. units (c) Temperature

Figure 6.3: Approximation of the evolution of standard deviation of populations and the temperature variable in the client–server model.

Figure 6.3 shows approximations of standard deviations in the client–server model. As demonstrated in Section 4.2, the ODE analysis is quite accurate in case of standard deviations of client and server populations, which are not dependent on the temperature variable, as depicted in Figure 6.3a. However, when applied to standard deviation of the population of air conditioning units, Figure 6.3b, and the temperature variable, Figure 6.3c, there are large quantitative and qualitative differences accumulated over time. The main source of error is the approximation of the minimum function from Equation 6.4 and Equation 3.10. Section 6.4.3 will discuss ways to improve the accuracy.

6.3.3 Relationship with accumulated populations

Continuous variables in a *h*PCTMC can be seen as generalisations of the accumulated populations from Chapter 5. Theorem 8 can be seen as a general form covering the cases of mean accumulated populations, Equation 5.7, higher-order moments of accumulated populations, Theorem 6, and higher-order moments of accumulated products of populations, Theorem 7.

The energy consumption of servers $\mathcal{A}_{\text{energy}}(t)$ in the client–server model, Equation 5.1, can be directly defined as a continuous variable:

$$\mathcal{Y}_{energy}(t) = \int_0^t S(u) \cdot r_S + S_g(u) \cdot r_{S_g} + S_b(u) \cdot r_{S_b} \mathrm{d}u$$

Table 6.2: Comparison between ODE systems used to calculate variance of the reward $\mathcal{A}_{\text{energy}}(t)$ using moments of accumulated populations and the *h*PCTMC framework. The variable $X(t)$ stands for any of the six populations in the client–server model and the variable $S(t)$, $S'(t)$ and for any of the three server populations.

	Accumulated populations		***h*PCTMC**	
Pop. means	$\mathbb{E}[X(t)]$	6	$\mathbb{E}[X(t)]$	6
Pop. second	$\mathbb{E}[X(t)X'(t)]$	21	$\mathbb{E}[X(t)X'(t)]$	21
Acc. means	$\mathbb{E}[\overline{S}(t)]$	3	$\mathbb{E}[\mathcal{Y}_{energy}(t)]$	1
Acc. second	$\mathbb{E}[\overline{S}(t)\overline{S'}(t)]$	6	$\mathbb{E}[\mathcal{Y}^2_{energy}(t)]$	1
Mixed	$\mathbb{E}[X(t)\overline{S}(t)]$	18	$\mathbb{E}[X(t)\mathcal{Y}_{energy}(t)]$	6
Total		54		35

The mean of $\mathcal{Y}_{energy}(t)$ can be approximated by a single mean-field ODE using Equation 6.4, whereas the mean of $\mathcal{A}_{\text{energy}}(t)$ is expressed in Equation 5.5 as a linear combination of three accumulated populations, each captured by an ODE.

In order to calculate variance of $\mathcal{A}_{\text{energy}}(t)$, we have to consider auxiliary ODEs approximating second-order moments of accumulated populations of the three server states, the second-order moments of the three combinations of pairs of accumulated server states, and mixed moments between each system population and each accumulated server population. In the case of variance of $\mathcal{Y}_{energy}(t)$, there is only a single second-order moment of the accumulated variable and only mixed moments between each system population and the accumulated variable. Table 6.2 gives an overview of the numbers of different ODE types in the system.

6.4 Convergence properties

In this section we will prove that, in the limit of large populations, a suitably rescaled *h*PCTMC model converges to its mean-field approximation. We present results analogous to those for the PCTMC case from Section 3.6. Given a *h*PCTMC $(\boldsymbol{X}(t), \boldsymbol{\mathcal{Y}}(t), \mathcal{C}, \boldsymbol{g}, \boldsymbol{x_0}, \boldsymbol{y_0})$, we construct a sequence of *h*PCTMCs $(\boldsymbol{X}^{(S)}(t), \boldsymbol{\mathcal{Y}}^{(S)}(t), \mathcal{C}^{(S)}, \boldsymbol{g}^{(S)}, \boldsymbol{x}_{\boldsymbol{0}}^{(S)}, \boldsymbol{y}_{\boldsymbol{0}}^{(S)})$. We assume that the population structure and difference vectors in $\mathcal{C}^{(S)}$ are the same as in $\mathcal{C}$, but that the rate functions $r_c^{(S)}$ and the accumulation functions $\boldsymbol{g}^{(S)}$ vary with S. The initial conditions for the S-th model in the sequence are given by $(S\boldsymbol{x_0}, S\boldsymbol{y_0})$. For each model in this sequence, we assume that the assumptions of Section 6.2 are satisfied so that all of the processes are well defined and write $\mathbb{X}^{(S)} \subseteq \mathbb{Z}_+^N$ for the reachable state space of the discrete component of the S-th process.

We assume further that the rate functions $r_c^{(S)}$ and the accumulation functions $\boldsymbol{g}^{(S)}$ satisfy the density dependent property from Section 3.6, that is they can be defined as functions of the scale S as follows:

$$r_c^{(S)}(\boldsymbol{x}, \boldsymbol{y}) := S r_c(\boldsymbol{x}/S, \boldsymbol{y}/S) \qquad c \in \mathcal{C}^{(S)}$$
$$\boldsymbol{g}^{(S)}(\boldsymbol{x}, \boldsymbol{y}) := S\boldsymbol{g}(\boldsymbol{x}/S, \boldsymbol{y}S)$$

We will consider the mean-field Equation 6.4 for the original *h*PCTMC $(\boldsymbol{X}(t), \boldsymbol{\mathcal{Y}}(t))$ and assume that the functions $\boldsymbol{f}$ and $\boldsymbol{g}$ satisfy local Lipschitz conditions uniformly with respect to t over any compact interval. Further, we assume that solutions to the mean-field equations exist globally. It is easy to see that due to the above definition of the rates and accumulations in the *h*PCTMC models in the sequence, the mean-field equations for these are the same as for the original *h*PCTMC, with the difference of initial conditions.

We define the rescaled process $(\bar{\boldsymbol{X}}^{(S)}(t), \bar{\boldsymbol{\mathcal{Y}}}^{(S)}(t))$, as:

$$\bar{\boldsymbol{X}}^{(S)}(t) := \boldsymbol{X}^{(S)}(t)/S \qquad \bar{\boldsymbol{\mathcal{Y}}}^{(S)}(t) := \boldsymbol{\mathcal{Y}}^{(S)}(t)/S$$

We require that there is some compact subset of $\mathbb{R}^N$ that contains all of the state spaces of the rescaled processes $\bar{\boldsymbol{X}}^{(S)}(t)$. We also assume that $\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}) \leq C(\|\boldsymbol{x}\| + \|\boldsymbol{y}\| + 1)$ for all $\boldsymbol{x} \in \mathbb{R}^N_+$, $\boldsymbol{y} \in \mathbb{R}^M$ for some $C \in \mathbb{R}_+$. Then by an application of Grönwall's lemma similar to that of Section 6.2, we have that for all $t \in [0, t_f]$, for some $t_f \in \mathbb{R}$ the rescaled stochastic processes and the mean-field approximations can be contained within a single compact set $\mathbb{S} \subset \mathbb{R}^{N+M}$ that is independent of S. Note that it is then only strictly necessary for the following theorem that $\boldsymbol{f}$ and $\boldsymbol{g}$ are defined on $\mathbb{S}$ rather than on the whole of $\mathbb{R}^{N+M}$. Finally, we require that $r_c^{(S)}(\boldsymbol{x}, \boldsymbol{y}) \leq D(\|\boldsymbol{x}\| + \|\boldsymbol{y}\| + S)$ for all $c \in \mathcal{C}$, $\boldsymbol{x} \in \mathbb{X}^{(S)}$, $\boldsymbol{y} \in \{S \cdot s : s \in \mathbb{S}\}$ where $D \in \mathbb{R}_+$ is independent of S.

6.4.1 First-order convergence

The following theorem, analogous to Theorem 3 shows that the rescaled processes converge in probability to the mean-field approximation of the original process.

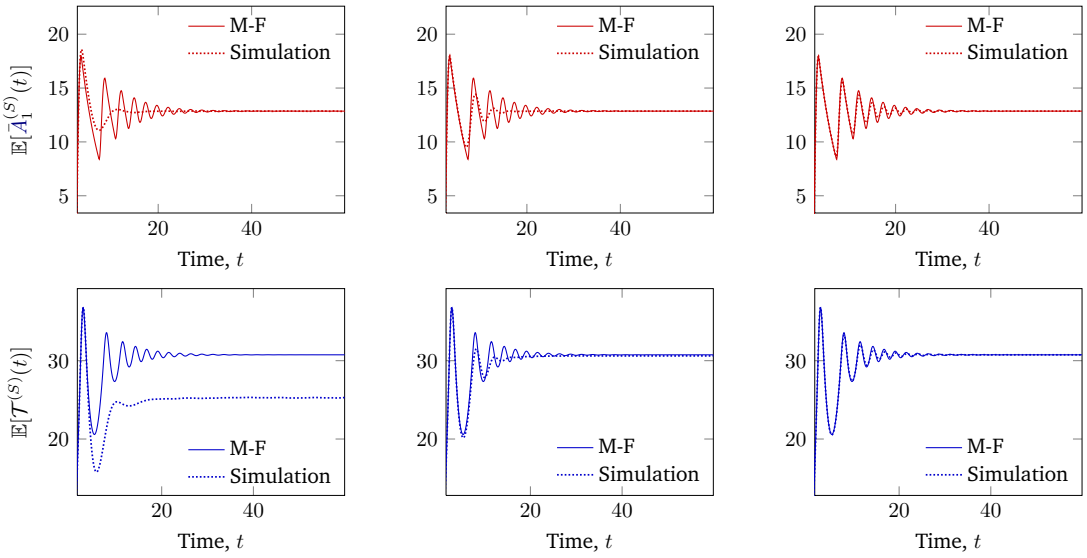
Theorem 9. Under the assumptions and setup given above, we have, for any $t_f > 0$ and $\epsilon > 0$:

$$\lim_{S\to\infty} \mathbb{P}\left\{\sup_{t\in[0,t_f]} \|\bar{\boldsymbol{X}}^{(S)}(t) - \boldsymbol{x}(t)\| > \epsilon\right\} = 0 \quad \lim_{S\to\infty} \mathbb{P}\left\{\sup_{t\in[0,t_f]} \|\bar{\boldsymbol{\mathcal{Y}}}^{(S)}(t) - \boldsymbol{y}(t)\| > \epsilon\right\} = 0$$

Proof. Due to Hayden. See Appendix C.1. □

> In the client–server model, scaling the number of agents by S, and, in particular, the number of servers, can be assumed to require a room approximately S times larger in volume than that of the original system. Therefore if the initial heat energy content of the room E_0 is scaled by S and the total heat energy content of the room is divided by S, the temperature as S increases is:
>
> $$\mathcal{T}^{(S)}(t) = \frac{c}{Sv}\mathcal{E}^{(S)}(t) \quad \text{and} \quad \mathcal{E}_0^{(S)} = SE_0$$

Theorem 9 requires a continuity assumption on the transition rate and accumulation functions in a *h*PCTMC. The rate functions $\lambda_{on}^{(S)}$ and $\lambda_{off}^{(S)}$ in the client–server model satisfy these requirements. Figure 6.4 observes the exact means (estimated from simulation) converging to the solutions of the mean-field equations.



(a) $S = 1$ (b) $S = 10$ (c) $S = 100$

Figure 6.4: Effect of scaling the system size on the first order mean-field approximation of air conditioning units population and the temperature variable in the client–server model.

6.4.2 Second-order convergence

In this section, we give a second-order Gaussian convergence result for the sequence of rescaled hPCTMC models, which will justify the use of second-order moment equation from Theorem 8 and motivate the use of the min-normal closure in Section 6.4.3. We maintain all of the notation of the previous section.

In addition to the main assumptions of this section, we assume that we can decompose the functions $\boldsymbol{f}$ and $\boldsymbol{g}$ such that

$$\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y}) = \sum_i \mathbf{1}_{\{(\boldsymbol{x},\boldsymbol{y})\in F_i\}}\boldsymbol{f}^i(\boldsymbol{x},\boldsymbol{y})$$
$$\boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}) = \sum_j \mathbf{1}_{\{(\boldsymbol{x},\boldsymbol{y})\in G_j\}}\boldsymbol{g}^j(\boldsymbol{x},\boldsymbol{y})$$

where $\{F_i\}$ and $\{G_j\}$ are finite collections of disjoint open sets in $\mathbb{R}^N \times \mathbb{R}^M$ such that for each i, respectively j, $\boldsymbol{f}^i(\cdot,\cdot)$, respectively $\boldsymbol{g}^j(\cdot,\cdot)$ is totally differentiable on $F^i \cap \text{int}(\mathbb{S})$, respectively $G^j \cap \text{int}(\mathbb{S})$, for all $t \in \mathbb{R}_+$ with uniformly continuous total derivative there. Then $\boldsymbol{f}(\cdot,\cdot)$ and $\boldsymbol{g}(\cdot,\cdot)$ have uniformly continuous total derivative on $\cup_i F_i \cap \text{int}(S)$ and $\cup_i G_i \cap \text{int}(\mathbb{S})$ respectively, which we write as $D\boldsymbol{f}$, $D\boldsymbol{g}$. We can extend Theorem 4 to hPCTMC:

Theorem 10. Fix $t_f > 0$. Assume that the set $\{t \in [0,t_f] : (\boldsymbol{X}(t),\boldsymbol{\mathcal{Y}}(t)) \notin \cup_i F_i \cap \cup_j G_j \cap \text{int}(\mathbb{S})\}$ has Lebesgue measure zero. Then for mutually independent standard Brownian motions $\{B_c(t) : c \in \mathcal{C}\}$, the following equations have a unique strong solution [e.g. 120, Theorem 6.30] such that

$(\boldsymbol{E}^X(t), \boldsymbol{E}^Y(t))$ defined below is jointly-Gaussian:

$$\boldsymbol{E}^X(t) := \int_0^t D\boldsymbol{f}(\boldsymbol{x}(s), \boldsymbol{y}(s)) \cdot (\boldsymbol{E}^X(s), \boldsymbol{E}^Y(s))^T \,\mathrm{d}s + \sum_{c \in \mathcal{C}} B_c \left(\int_0^t f_c(\boldsymbol{x}(s), \boldsymbol{y}(s)) \,\mathrm{d}s \right) \boldsymbol{\delta}_c$$

$$\boldsymbol{E}^Y(t) := \int_0^t D\boldsymbol{g}(\boldsymbol{x}(s), \boldsymbol{y}(s)) \cdot (\boldsymbol{E}^X(s), \boldsymbol{E}^Y(s))^T \,\mathrm{d}s$$

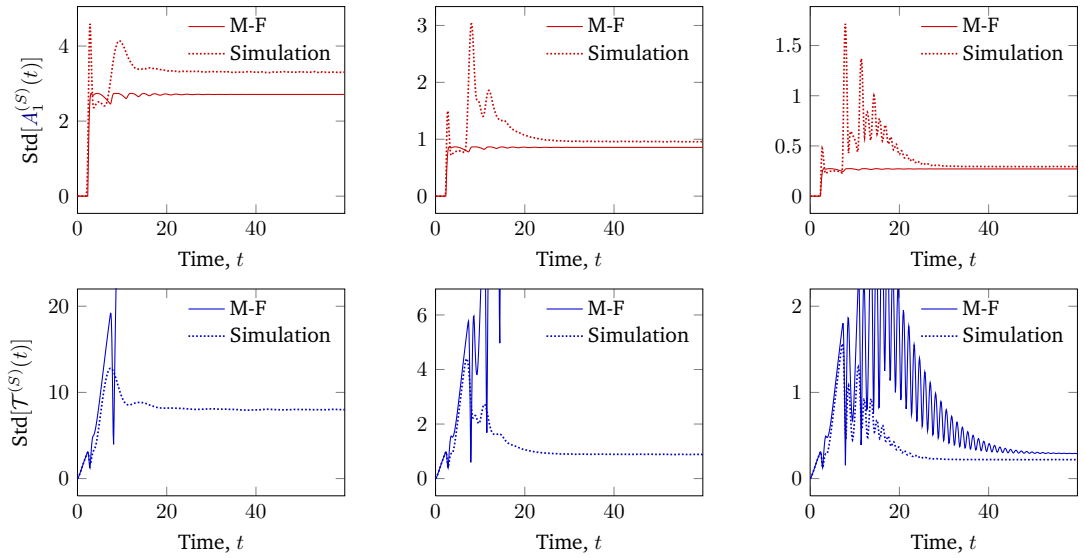
Furthermore,

$$\left(\frac{\boldsymbol{X}^{(S)}(t) - S\boldsymbol{x}(t)}{\sqrt{S}}, \frac{\boldsymbol{Y}^{(S)}(t) - S\boldsymbol{y}(t)}{\sqrt{S}} \right) \Rightarrow (\boldsymbol{E}^X(t), \boldsymbol{E}^Y(t)) \qquad \text{as } S \to \infty$$

where the convergence is weak on $D([0, t_f]; \mathbb{R}^{n+m})$ endowed with the uniform topology [e.g. 32]. Informally, this is 'uniform convergence in distribution over $[0, t_f]$'.

Proof. Due to Hayden. See Appendix C.1. □

The assumptions of Theorem 10 apply to the client–server model. Figure 6.5 shows convergence of standard deviations of the active air conditioning units population and the temperature variable. Unlike in the mean case, the convergence seems to be much slower and the approximations are not very accurate even at the highest scale shown where $S = 100$. In the next section, based on the Gaussian assumption justified by Theorem 10, we use the min-normal closure from Section 4.4 to improve the standard deviation approximations.



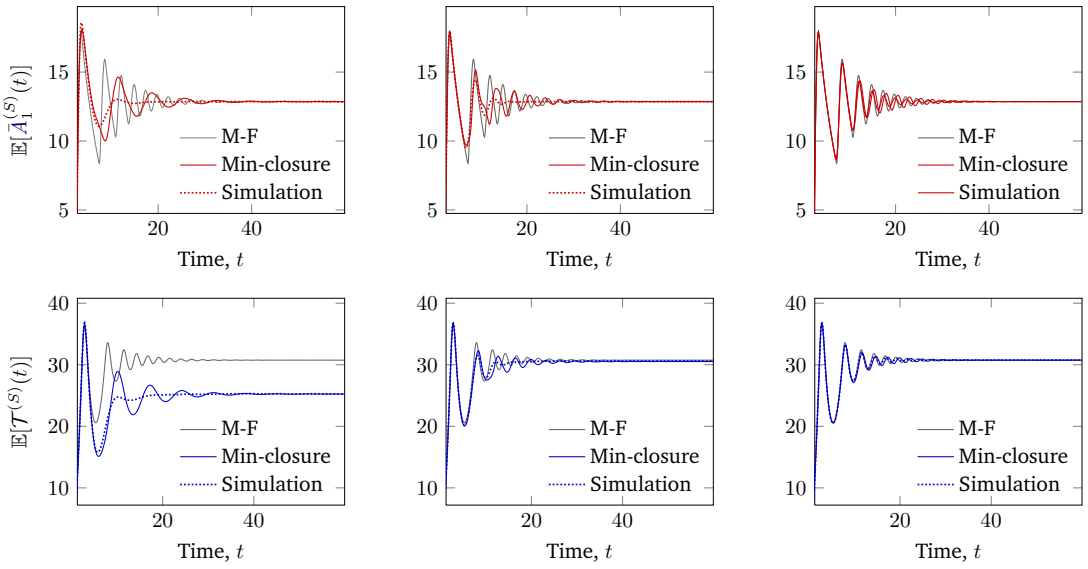
(a) $S = 1$ (b) $S = 10$ (c) $S = 100$

Figure 6.5: Effect of scaling on the mean–field approximation of standard deviation of air conditioning units population and of the temperature variable in the client–server model.

6.4.3 Normal approximations

Theorem 10 suggests that both the discrete and continuous components $\boldsymbol{X}(t)$ and $\boldsymbol{\mathcal{Y}}(t)$ of a *h*PCTMC can be approximated by a jointly Gaussian process for sufficiently large populations. The main source of error in the mean and standard deviation approximations, Figure 6.4 and Figure 6.5 respectively, was the approximation of the expectation of a minimum function on the right-hand side of the mean and second-order moment ODEs. We can apply the min-normal closure from Section 4.4 to the ODEs for moments of agent populations as well as moments of the continuous variables. For terms involving a product of a rate and a continuous variable, we use the approximation previously used for accumulated populations, Equation 5.10.

Figure 6.6 compares simulation estimates with numerical solutions to the mean-field equations and with solutions to the new set of equations obtained by replacing occurrences of the minimum function according to the min-normal closure. We see that this results in significant improvements in accuracy.

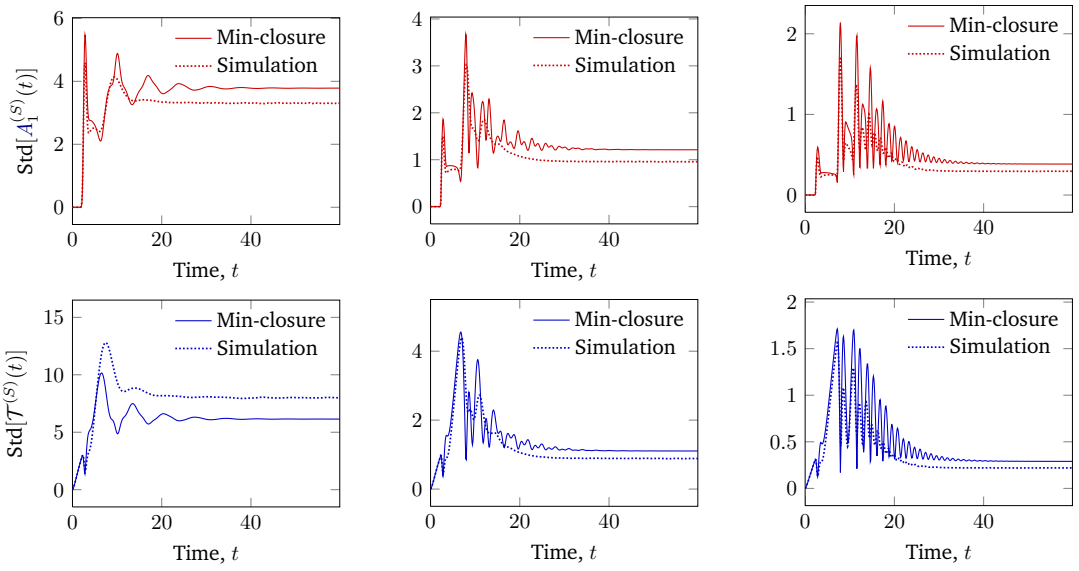


(a) $S = 1$ (b) $S = 10$ (c) $S = 100$

Figure 6.6: Effect of scaling on the min-closure approximation of mean air conditioning units population and the temperature variable.

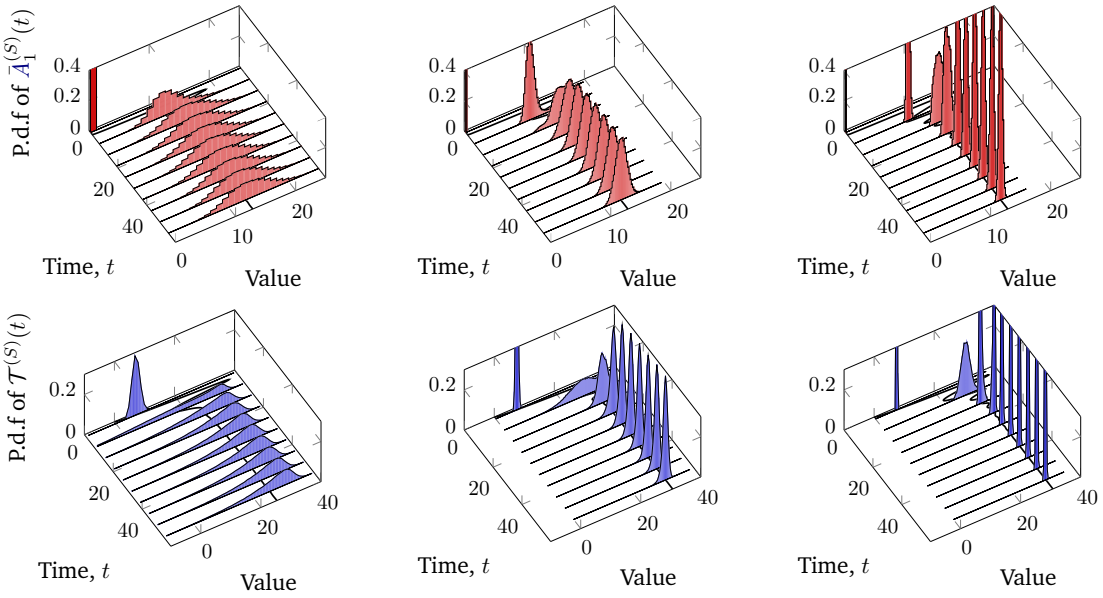
Figure 6.7 further shows that the min-normal moment closure can result in an accurate approximation of the standard deviation of the temperature variable even at relatively low scales of the system.

Figure 6.8 shows the distribution of the population of air conditioning units and the temperature over time (obtained from stochastic simulation), with means and variances shown in Figure 6.6 and Figure 6.5. At the lowest scale, the temperature variable is clearly skewed away from a normal distribution. This means that the approximation used by the min-normal closure, Equation 4.2, is not necessarily accurate and results in the errors seen in the respective plots of the mean and standard deviation. As the system scale increases, both the temperature variable and the active air conditioning population get closer to a normal distribution, in agreement with Theorem 10.



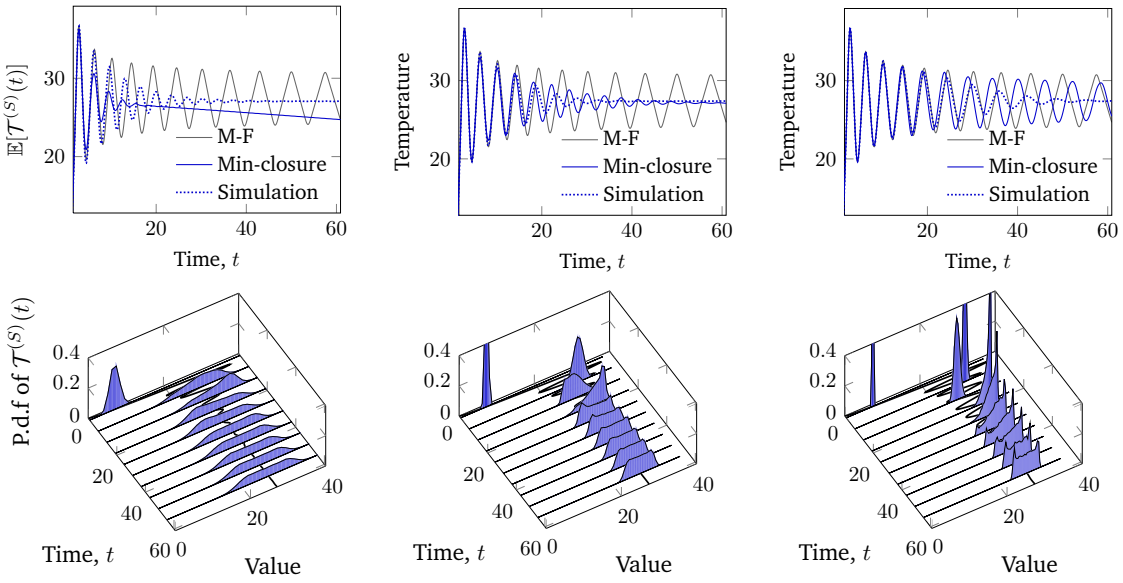
(a) $S = 1$ (b) $S = 1$ (c) $S = 1$

Figure 6.7: Effect of scaling on the min-closure approximation of standard deviation of air conditioning units population and the temperature variable. The closure significantly improves the accuracy over the mean-field approximation, Figure 6.5.



(a) $S = 1$ (b) $S = 10$ (c) $S = 100$

Figure 6.8: Distribution of the number of active air conditioning units (top row) and the temperature variable (bottom row), as the system evolves over time (x axis). Each column shows the distribution for an increased scale of the system.



(a) $S = 1$ (b) $S = 10$ (c) $S = 100$

Figure 6.9: Effect of scaling on the mean temperature for the client–server model with two thresholds. The bottom row shows the probability density function of the temperature at each point in time as the system evolves.

6.4.4 Limitations – speed of convergence

Figure 6.6 shows the convergence of the mean air conditioning population and temperature variable in the client–server model. The mean-field approximations and the means from simulation converge to each other as the system gets scaled. In this particular case, the accuracy of the approximation is very high for the whole time interval $t \in (0, 60)$ at the largest shown scale $S = 100$.

Theorem 9 only describes the relationship between the joint process $(\boldsymbol{X}^{(S)}(t), \boldsymbol{Y}^{(S)}(t))$ and the mean-field approximation $(\boldsymbol{x}(t), \boldsymbol{y}(t))$ in the limit as $S \to \infty$. The accuracy at particular values of the scale S and for specific final times t_f cannot be predicted and heavily depends on the model. For example, the client–server model can be extended with a second threshold in the temperature control. In addition to air conditioning units becoming active with the proportional rate in Equation 6.3, we can define the rate of switching off to be proportional to a distance from a second, lower, temperature threshold

$$\lambda_{off}(\mathcal{T}) = (t'_{thresh} - \mathcal{T})^{+} r_{off} \tag{6.6}$$

Setting $t'_{thresh} = 25$ and leaving the rest of the system identical, we get significantly different behaviour in the temperature variable and its convergence properties.

Figure 6.9 compares the mean temperature obtained from the mean-field technique and the normal approximations with the mean from simulation. The time interval where the approximations are accurate grows slowly with the scale S and even at the largest shown scale, the approximations do not agree with the simulation mean after $t \approx 30$. This can be explained by looking at the corresponding distribution of the temperature variable (the bottom row of Figure 6.9). Before

hitting any of the thresholds is very likely, the temperature can be approximated by a normal random variable. However, as hitting the thresholds becomes more likely, the temperature starts concentrating around the thresholds and reaches a bi-modal distribution.

6.5 Worked example

In this section we demonstrate the *h*PCTMC formalism and the efficient ODE analysis on a larger example of a heterogeneous computing cluster. Similar to the client–server model, we consider a high level abstraction of the system. We assume that there are two types of servers in the cluster — ones with low (class A) and ones with high power consumption (class B), respectively. Clients in the system submit two types of jobs — with low (type 1) and high loads (type 2) on the servers. As in the client–server model, we include air conditioning units that maintain the ambient temperature in the room. Additionally, servers are capable of entering a sleep mode in case the temperature increases above a threshold. Unlike in the case of the client–server model where the client and server agents in the discrete state space were unaffected by the continuous variables, this will result in an *h*PCTMC with a complete interdependence between the discrete agents and continuous variables.

We use the GPEPA process algebra to concisely describe the *h*PCTMC model ($j \in \{A, B\}$ is a server class and $i \in \{1, 2\}$ is a job type):

$$Client \stackrel{def}{=} \sum_{i=1}^{2} (queue_i, r_{q,i}).Job_i \qquad Server^j \stackrel{def}{=} \sum_{i=1}^{2} (service_i^j, r_{service,i}).Server_i^j + (sleep, \lambda_{sleep}(t)).Server_{sleep}^j$$

$$Job_i \stackrel{def}{=} (service_i, r_{service_i}).Client \qquad Server_i^j \stackrel{def}{=} (reset, r_{reset}).Server^j$$

$$Server_{sleep}^j \stackrel{def}{=} (wakeup, r_{wakeup}).Server^j$$

$$\Big(\mathbf{Servers}\{Server^A[n_{SA}] || Server^B[n_{SB}]\} \parallel \mathbf{Aircon}\{Aircon_0[n_A]\}\Big) \underset{\{service_i | 1 \le i \le 4\}}{\bowtie} \mathbf{Clients}\{Client[n_C]\}$$

with rates $\lambda_{off}(t) = r_{on}$ and

$$\lambda_{sleep}(t) = (\mathcal{T}(t) - t_{sleep})^+ \cdot r_{j,sleep} \quad \text{and} \quad \lambda_{on}(t) = (\mathcal{T}(t) - t_{thresh})^+ \cdot r_{on}$$

where temperature is defined as in Equation 6.2 and the energy variable is

$$\mathcal{E}(t) = E_0 + \int_0^t \sum_j (S^j(u)c_{j,s} + S_{sleep}^j(u)c_{j,sl} + S_1^j(u)c_{j,1} + S^s(u)c_{j,2}) - A_1(u)c_a \mathrm{d}u$$

for some constants $c_{j,s}, c_{j,sl}, c_{j,1}, c_{j,2}, c_a$.

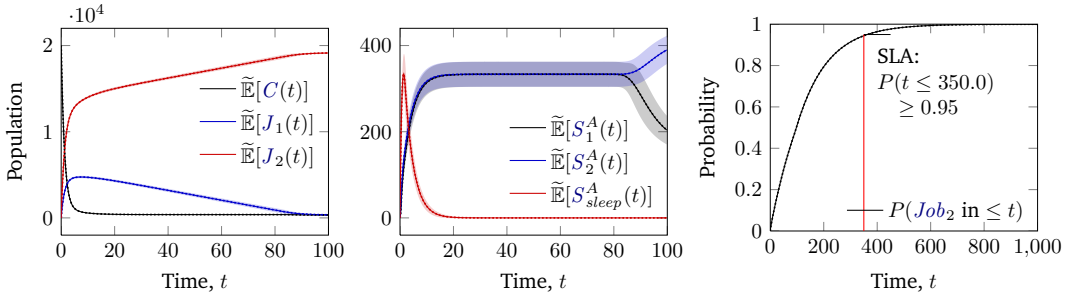
Additionally, we query the model to calculate SLA derived passage times as described in Section 6.6. We compute the time until an individual client executes its first high load job. Such

measures are often used when expressing SLAs. The example shows how the presented framework can be used to study the trade-off between SLA satisfaction and the energy efficiency of the system. An increasingly common metric assessing energy efficiency of data centres is the *Power Usage Efficiency (PUE)* metric [157], calculated as the ratio between the total energy consumption and the energy used by the servers. That is, PUE of 1 represents a perfectly efficient data centre. In the above model, we can model the total energy consumption as a continuous variable:

$$\mathcal{P}(t) = \int_0^t \sum_j (p_{j,sl} S^j_{sleep}(u) + p_{j,s} S^j(u) + p_{j,1} S^j_1(u) + p_{j,2} S^j_2(u)) + p_a A_1(u) \mathrm{d}u$$

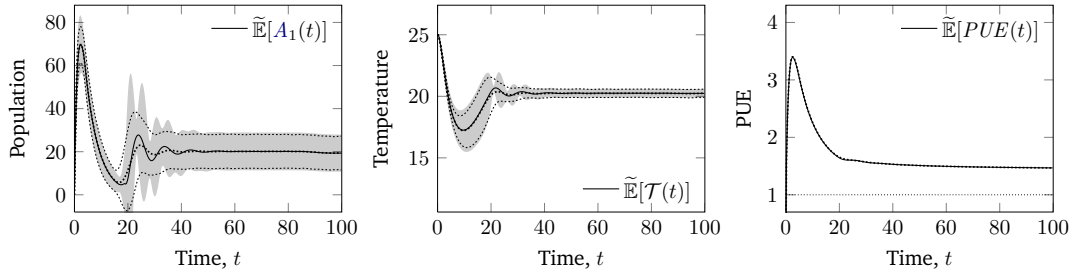
for some constants $p_{j,s}, p_{j,sl}, p_{j,1}, p_{j,2}, p_a$.

The quantity $\mathcal{U}(t)$ represents the energy used for computation and is defined as $\mathcal{P}(t)$, omitting the contribution of the air conditioning units and the servers in the sleeping state. To obtain an approximation of the mean PUE, we compute $\widetilde{\mathbb{E}}[\mathcal{P}(t)]/\widetilde{\mathbb{E}}[\mathcal{U}(t)]$ for sufficiently large t (1000 in the examples below).



(a) Mean client populations (b) Mean server populations (c) Passage-time CDF

Figure 6.10: Means of client–server populations and passage-time CDF in the computing cluster model. The shaded regions are 1.95 standard deviations around the respective means.



(a) Active aircon. units (b) Temperature (c) PUE

Figure 6.11: Mean population of active air conditioning units and mean temperature in the cluster model. The approximation of standard deviation in figure (b) was obtained by applying the normal min closure from Section 6.4.3.

Figure 6.10 shows the mean populations of client and server-A agents and the passage-time CDF as obtained by the mean-field analysis. Figure 6.11 shows the mean population of air conditioning units, its effect on the mean controlled temperature and the PUE of the system. The used values of all the model parameters are shown in Table 6.3.

Table 6.3: Values of rate and initial population parameters used in the worked example, Figures 6.10, 6.11, 6.12. The constants $p_{B,\cdot}$ are set as the respective $p_{A,\cdot}$ constants multiplied by 1.7 and the heat constants $c_{\cdot,\cdot}$ are set as the corresponding $p_{\cdot,\cdot}$ constants multiplied by a conversion factor 7.71×10^{-6}.

n_C	n_S	n_A	$r_{q,1}$	$r_{q,2}$	$r_{s,1}$	$r_{s,2}$	r_{reset}	r_{wakeup}	
20000	1000	100	0.2	0.5	0.2	0.2	0.2	0.3	
r_{on}	r_{off}	T_0	t_{thresh}	t_{sleep}	$p_{A,s}$	$p_{A,sl}$	$p_{A,1}$	$p_{A,2}$	r_{cool}
0.2	0.2	25	20	23	10	1	30	37.5	0.026

One benefit of mean-field analysis is the relatively low computational cost of numerically integrating the mean-field equations. This allows a rapid evaluation of a large number of system configurations, such as in the energy–performance trade-off case study in Section 5.5.1. For example, we can look at the relationship between the two temperature thresholds t_{thresh} and t_{sleep} that specify when the air conditioning units start contributing to cooling and servers switch to sleep mode, respectively. We fix the server threshold at 23 units and search for the best air conditioning threshold. Our target measure to minimise will be the PUE in steady state of the system and the constraints are given by requiring satisfaction of the above SLA. Figure 6.12 explores a range of system configurations with the number of servers of each type $n_S = n_{SA} = n_{SB}$ varying between 50 and 1500 and the threshold t_{thresh} varying between 20 and 26 units.

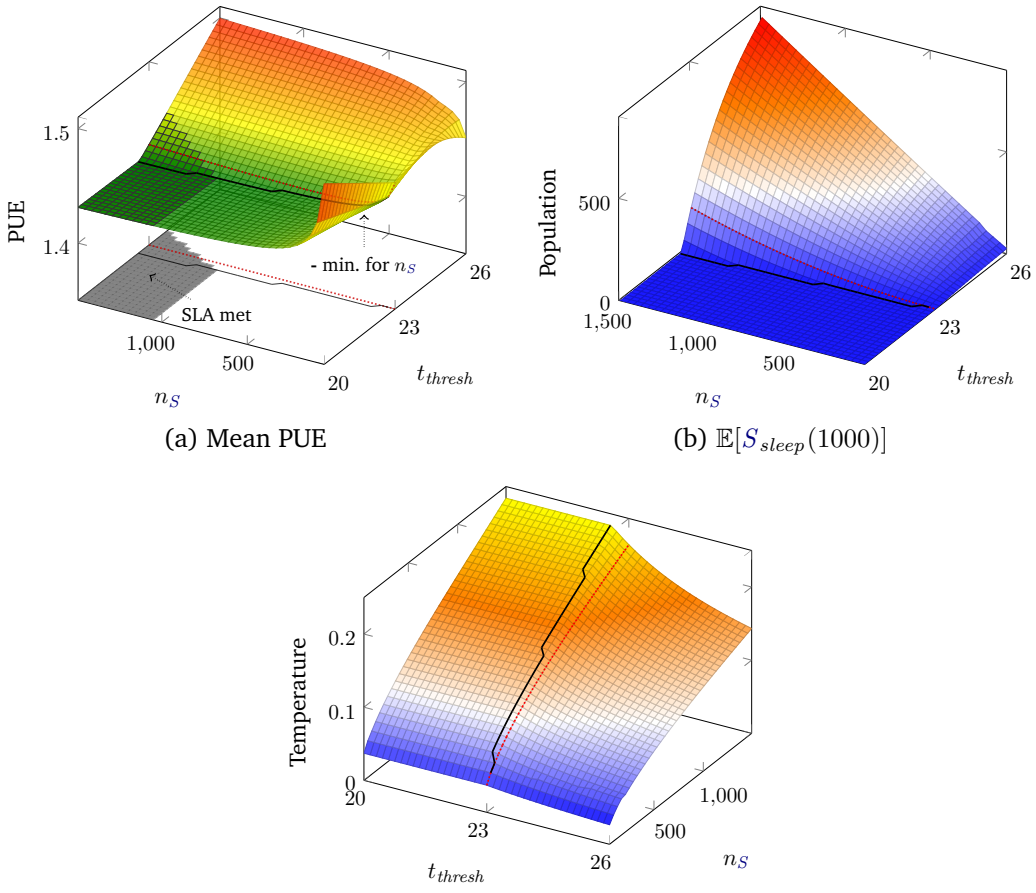
Figure 6.12a shows the mean steady-state PUE for each configuration. For each size of the computing cluster given by a value of n_S, there is an optimal value of t_{thresh} achieving a minimal PUE metric. These thresholds and the corresponding optimal PUE values are shown by the thick solid line.

It can be seen that this is slightly below the server threshold, shown as the red dotted line. For example, for $n_S = 850$, the value of t_{thresh} achieving the optimal PUE is 22.7. The SLA is achieved only when there are sufficiently many servers in the system, shown as the darker region on the surface plot. Figure 6.12b shows that the optimal PUE line minimises the number of sleeping servers, while keeping the air conditioning units as lightly loaded as possible. Figure 6.12c shows that line of minimum PUE separates the region with maximal standard deviation of the temperature variable.

Figure 6.13 shows the absolute error in Figure 6.12 as compared to a large number of stochastic simulations. For the mean PUE and number of sleeping servers, figures Figure 6.13a and Figure 6.13b, the error would be barely visible on the graphs. For figure Figure 6.13c, the ODEs consistently under-estimate the standard deviation of the temperature. We note that the simulations used to evaluate these errors took several days of CPU time. Although we used parallel solution features of GPA and were able to distribute the load on hundreds of machines, such analysis would not be feasible in practical applications.

6.6 Time-inhomogeneous models

In this section we show how to use the *h*PCTMC framework to model time-inhomogeneous behaviour and how to adapt the passage time results for GPEPA to verify the validity of SLAs at different points in time. Time can be simply captured by a continuous variable with a unit



(a) Mean PUE

(b) $\mathbb{E}[S_{sleep}(1000)]$

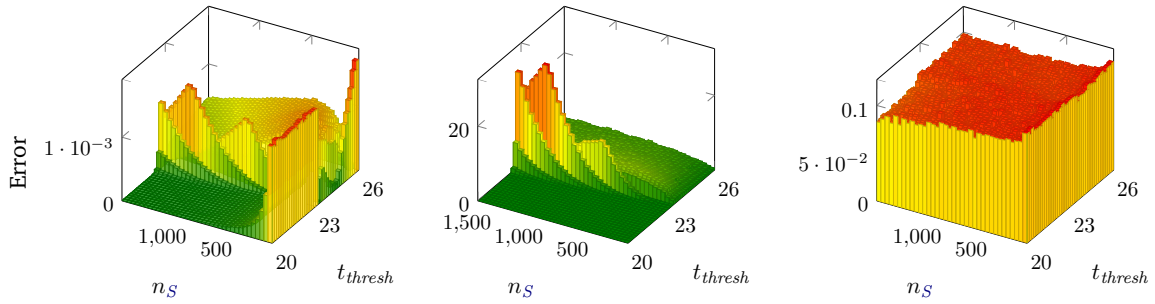
(c) Std$[\mathcal{T}(1000)]$

Figure 6.12: Effect of varying the cooling threshold and the number of servers on the steady state PUE metric and the number of servers in sleeping state. For each initial server population n_S, the thick black line shows the threshold under which the minimum PUE is achieved.

accumulation, $t = 0 + \int_0^t 1 \mathrm{d}s$. Rate and accumulation functions can depend on this variable in the same way they would for example depend on the temperature variable in the client–server example.

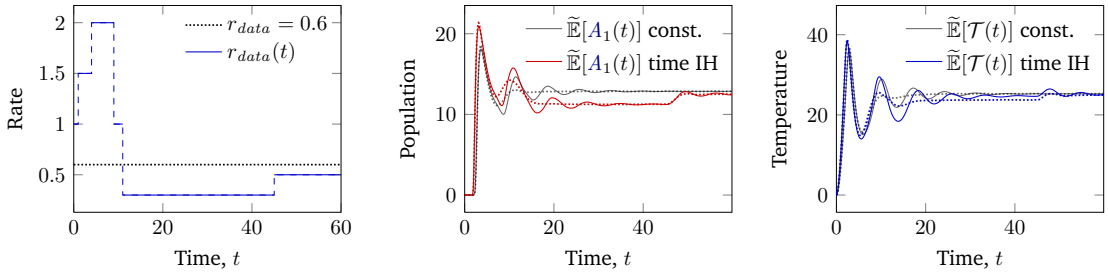
When modelling large-scale computer systems, it is often not suitable to assume that the system is stationary. For example, web applications can experience load that significantly varies over time, as shown in Figure 6.1. Similarly, data transfer rates can depend on external load on the network. For example, in the example client–server model, we have assumed that transfer between clients and servers happens at a constant rate r_{data}. However, if this system is placed in a larger network, it might happen that this rate is not constant over a given time period. The *h*PCTMC framework allows this rate to vary as a function of time t, $r_{data}(t)$. For example, we could use historical data to estimate the rate at some given points in time. The regularity conditions in Section 6.2.2 allow jumps in the rate functions and so we can extend the rate parameter to a continuous time interval as a piecewise constant function.

Figure 6.14a shows an example. The rate drops from around 1.5 requests per unit of time to 0.3 requests. This results in a lower number of active servers compared to the previous case with the rate constant at 0.6 requests and therefore in a lower heat generation and so a lower number of



(a) PUE (b) $\mathbb{E}[S_{sleep}(1000)]$ (c) Std$[\mathcal{T}(1000)]$

Figure 6.13: Absolute error in the plots in Figure 6.12, as compared to stochastic simulation.



(a) Time-inhomogeneous rate (b) Mean air conditioning population (c) Mean temperature

Figure 6.14: Client–server model with a time dependent rate of data transfer $r_{data}(t)$.

active air conditioning units, as seen in Figure 6.14b. The temperature is maintained around the desired threshold, shown in Figure 6.14c.

6.6.1 Dynamic SLA verification

In the client–server model, and in models of computer systems in general, a useful metric is the time it takes an individual client to go through a single data transfer. Usually, these metrics are part of service level agreements (SLA), which specify the minimum probability with which a client request has to be answered within a given maximum processing time. For example, in the client–server model, we can say that each client has to receive the data from servers within 10 time units at least 80% of the time. It is possible to derive this quantity by the ODE analysis[4], as described in Section 3.5. The state space of the model is extended with states that remember whether the data has been transmitted. The passage-time probability can be derived from a suitable expression involving first-order moments obtainable from the ODE analysis. Often, the service specification is not as simple as data transmission and can involve further requirements, such as the absence of failure or a pre-defined required sequence of sub-services. The *Unified Stochastic Probes* formalism (USP) [103] allows a large class of complex passage-times to be specified. These get translated into *probe* agents which can be composed with the original model and used to compute the passage-time probabilities within the ODE analysis framework.

If the system is considered to be stationary, the SLAs only have to be verified once – each request is assumed to arrive at the system under the same conditions. However, if the service characteristics change over time, care has to be taken to make sure that the SLAs are maintained

throughout the time interval of interest. The generated probes from the USP formalism allow description of passage times that are triggered after a given sequence of events. It is possible to use time-inhomogeneous rates to modify a probe so that it starts computing the passage-time probability only after an exact time period t_a. We can multiply the rate of the initial probe transition with an auxiliary inhomogeneous function

$$r_{t_a}(t) = \begin{cases} 0 & \text{if } t < t_a \\ 1 & \text{otherwise} \end{cases}$$

to disable the passage-time computation before t_a.

Figure 6.15 shows example passage-time distributions in the client–server model. The first CDF, corresponding to the time it takes to obtain the data for an individual client arriving at $t = 0$ can be obtained using the original techniques by Hayden et al. [4]. The second CDF, corresponding to the passage time of a client starting at $t_a = 20$ units, is computed with the time-inhomogeneous function above. If the SLA was to finish within 10 units with probability at least 0.8, the decrease in the data transfer rate would result in a violation at a later time.

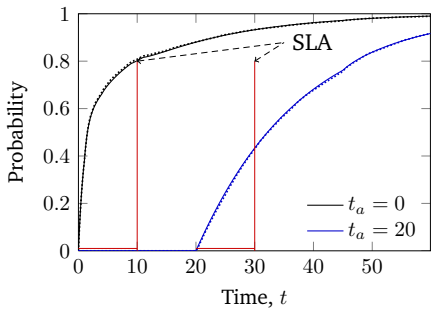


Figure 6.15: Dynamic SLA verification in the time-inhomogeneous client–server model. The figure shows the CDF of the time it takes an individual client to finish a data transfer, calculated with the mean-field analysis as an individual passage time. Two CDFs are shown - one for the transfer starting at time $t_a = 0$ and at $t_a = 20$. The red lines represent an SLA requirement to finish 10 time units after the start with probability at least 0.8.

One limitation is that only a finite number of starting times t_a can be queried. Each passage-time computation requires a solution of the extended system. The efficient nature of the ODE analysis allows a large number of starting times to be queried, but cannot guarantee the SLA for passage times starting at intermediate times. However, by inspecting the variations in the workload over time, it is often possible to select the starting times to cover the heaviest loads.

6.6.2 Worked Example

In this section we present an example of a hypothetical distributed computing system with time-inhomogeneous external arrivals with rate obtained from real data. We will use a widely analysed dataset of all the website accesses of the World Cup 1998 website [137]. Figure 6.1a shows how the website was accessed during its lifetime.

We start with a simple model abstracting the real system. We assume that the requests arrive at a time-inhomogeneous rate $\lambda(t)$. We model the system as a multi server queue. Each server can

process requests at rate μ. To save energy during periods of low load, the servers can switch to an idle state, with rate proportional to the excess number of servers above the current number of requests in the system. Additionally, we model air conditioning units that have to maintain a given temperature in the system. If the temperature exceeds a critical level T_{fail}, the servers start being prone to failures.

We can describe the resulting PCTMC in the notation from Equation 3.1:

$$\begin{aligned}
\emptyset &\to \mathit{Request} && \text{at rate } \lambda(t)\\
\mathit{Server} + \mathit{Request} &\to \mathit{Server} && \text{at rate } \min(R, S)\cdot\mu\\
\mathit{Server} &\to \mathit{Server}_{idle} && \text{at rate } (S-R)^{+}\cdot r_{down}\\
\mathit{Server}_{idle} &\to \mathit{Server} && \text{at rate } S_i\cdot r_{up}\\
\mathit{Server} &\to \mathit{Server}_{sleep} && \text{at rate } (\mathcal{T}-t_{fail})^{+}\cdot r_{fail}\\
\mathit{Aircon}_{off} &\to \mathit{Aircon}_{on} && \text{at rate } A_{off}\cdot(\mathcal{T}-t_{thresh})^{+}\cdot r_{on}\\
\mathit{Aircon}_{on} &\to \mathit{Aircon}_{off} && \text{at } A_{on}\cdot r_{off}
\end{aligned}$$

The accumulated variable for temperature $\mathcal{T}$ is defined in a similar way to the previous case study and we also include an accumulation capturing the total consumed power. The initial number of servers is n_S and air conditioning units n_A. The initial temperature is equal to the desired threshold $t_{thresh} = 25$ degrees.

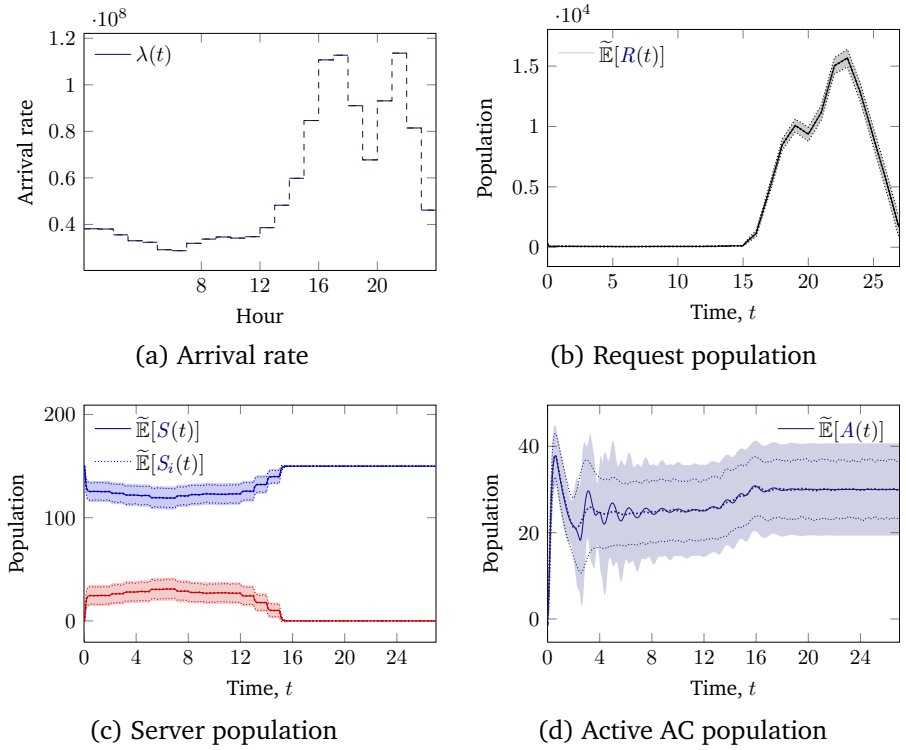
Similar to the previous case study, a question of interest would be to find the optimal number of servers n_S that minimises the operational costs of the system while coping with an expected workload. In this case, we will assume that the system will have to be employed to cope with a workload similar to that of the World Cup 1998 website, as shown in Figure 6.1.

Table 6.4: Values of rate and initial population parameters used in the time-inhomogeneous worked example, Figures 6.16, 6.18, 6.19. We obtained the arrival rate $\lambda(t)$ from the World Cup 98 data [137] available at `http://ita.ee.lbl.gov/html/contrib/WorldCup.html`. For illustration purposes, we have rescaled the arrival rate by a factor of 10^{-5}.

n_S	n_A	μ	r_{up}	r_{down}	r_{fail}
150	50	50.0	10.0	5.0	1.0
r_{on}	r_{off}	T_0	t_{thresh}	T_{fail}	
2.0	0.5	25	25	35	
p_S	p_{idle}	p_A	h_{cool}	h_S	h_{idle}
1.0	0.2	0.2	5.0	1.0	0.1

We assume a periodic workload and take an average busy day as representative of what the system has to cope with. To make the model simple, we fit a simple arrival process to match the average number of arrivals for each hour in the day. If we set the time unit in the PCTMC model as one hour, we can set the arrival rate $\lambda(t)$ to be a piecewise constant function with values equal to the average number of accesses during each hour. Figure 6.16 shows the arrival rate and a sample behaviour of the system. For this particular configuration, all the servers are occupied during the busy time between the hours 16 and 24 Figure 6.16c. The air conditioning units reflect

this and more are active during this time period Figure 6.16d. Table 6.4 lists the used parameter values for the examples in this section.



(a) Arrival rate (b) Request population

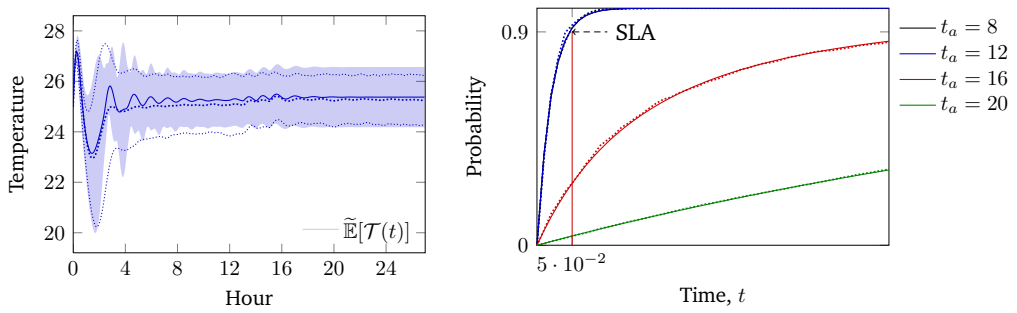
(c) Server population (d) Active AC population

Figure 6.16: The external arrival rate piecewise continuous $\lambda(t)$ and evolution of population means in the inhomogeneous multiserver model.

The increased load affects the service quality for each individual request. This can be captured by the time between arrival and end of processing of the request. Using the method described in Section 6.6, we can efficiently compute the CDFs of processing times for requests arriving at various fixed times. We set these to $8, 12, 16$ and 20 (the low computational cost of the ODE analysis would allow us to keep track of many more). Figure 6.17b shows the passage time probabilities. For this configuration, the high load between the hours 16 and 24 results in a much slower processing time for requests arriving at times 16 and 20. Figure 6.17a shows the mean temperature in the system. It can be seen that this is close to the controlled threshold and rarely reaches the thresholds that triggers server failure.

A possible SLA can guarantee that each request gets processed within 0.05 hours at least 90% of the time. We would want to find the system configuration that is able to satisfy this. The low computation cost of mean-field analysis allows us to explore a large number of system configurations. We will perform the analysis with large number of different values of n_S and n_A. Figure 6.18 shows the passage time probabilities for different system combinations. As expected, requests arriving during busier times can only be satisfied with respect to the SLA when there are sufficiently many servers in the system. In each case, the number of air conditioning units has to be sufficient to maintain the temperature below the failure level.

The total daily power consumption can be computed at the same time when looking for the passage time probabilities. Figure 6.19 shows the daily power consumption and the absolute difference between the results from mean-field analysis and simulation. As expected, the optimal



(a) Mean temperature

(b) Passage times for different arrival times

Figure 6.17: Temperature and multiple passage time CDFs in the inhomogeneous multiserver model. Figure (b) shows the CDFs of time to completion for clients arriving at different points in time of the system. The red line shows an SLA level of 90% guarantee to finish within 0.05 hours.

number of servers which minimises the energy consumption is the smallest value of n_S in the intersection of feasible regions for the passage times with different request arrival times.

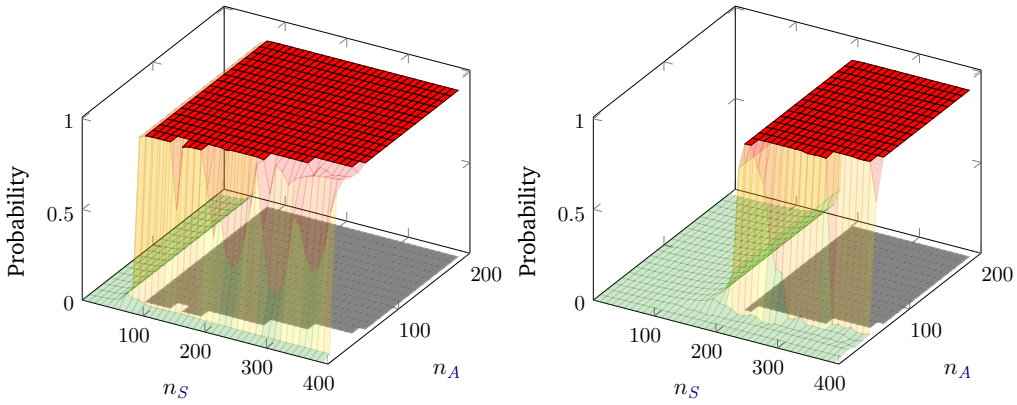
6.7 Conclusion

In this chapter, we defined hybrid PCTMCs, *h*PCTMCs, which generalise the rewards from Chapter 5 and allow continuously accumulated variables to be included in the state space of models. The variables can appear in transition rates of the discrete component of the system and thus allow feedback loops controlling the evolution of the continuous variables. We have shown how to use a continuous variable to represent time and thus allow time-inhomogeneous behaviour. We have shown how to efficiently extract passage time measures in a system with non-stationary behaviour.

One of the main advantages of the framework is that it enables efficient ODE analysis of the resulting models. This gives access to various metrics in the models without having to resort to computationally expensive stochastic simulation. Although the ODE analysis is an approximation, we have proved convergence results that guarantee the approximation error to decrease as the systems get larger. We have also used a moment closure based on the normal distribution from Section 4.4 that can improve the accuracy at a much lower computational cost than simulation.

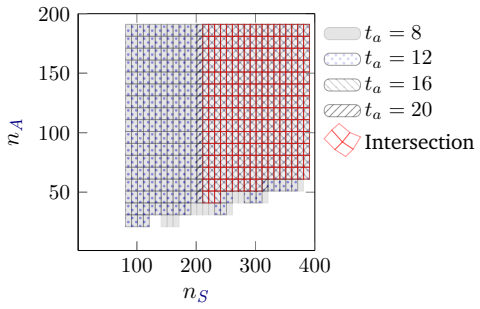
We have demonstrated the framework on an example of a large scale many server system under a time-varying load. We have shown how to obtain the arrival rates from real data and used an available website access data in our example. An important advantage of the ODE analysis techniques is the low computational cost that can be used to explore a large number of different system configurations. The framework thus ultimately allows us to experiment with a large number of system configurations and find one where a client SLA specification is met while server running temperature is maintained at a desired level and the total energy consumption of the system is minimised.

All of the numerical results in this chapter were produced using an implementation of the techniques in an extension to the GPA tool, described in Chapter 8.



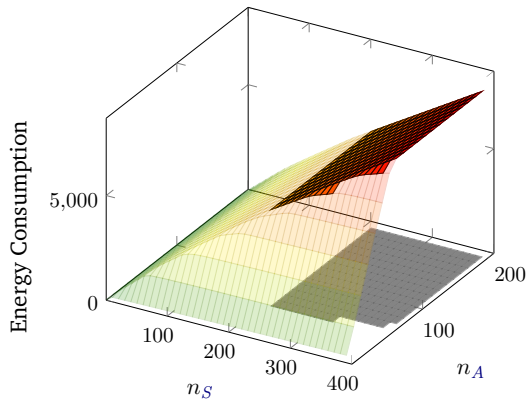
(a) $\mathbb{P}(T_f \leq 0.05 \mid t_a = 8)$

(b) $\mathbb{P}(T_f \leq 0.05 \mid t_a = 20)$

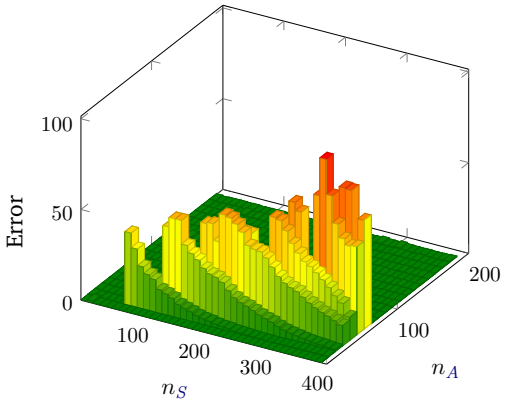


(c) Feasible regions for the SLA at different request arrival times

Figure 6.18: Request processing time probabilities in a large number of configurations of the multiserver example. Figure (a) shows the probabilities for requests arriving at time 8 and figure (b) at time 20. The shaded surface is the feasible region where the probability is greater than the one given by the SLA. Figure (c) shows the feasibility regions for all 4 request arrival times and their intersection.



(a) Mean energy



(b) Error

Figure 6.19: Daily power consumption for different configurations of the multiserver system.

Chapter 7

High-level specification of transactions

Key contributions		
GPEPA with channels for transaction layer over GPEPA	7.2	[10]
Large scale computing cluster case study	7.4	[10]

7.1 Introduction

In previous chapters of this thesis, we presented PCTMC models of systems consisting of a large number of interacting agents, where populations represent the number of agents in different states. This is a natural abstraction and allows intuitive descriptions of systems from various domains. One drawback of this approach is that all the interactions among a particular set of agents can last only for the duration of a single system transition. For example GPEPA models do not maintain a persistent session information between agents from different groups. In the client–server model from Section 3.2.3, a client requests data from a server in two stages. In the presented version of this model, there is no guarantee that the server responding to the client request is the same as the one providing the data. In various applications, this is a crucial feature of the modelled system - the agents behave according to an agreed protocol enter a longer lasting *transaction*, throughout which they interact exclusively with each other. For example, in virtualised systems, each server is capable of running a number of virtual machines (VMs). This can be captured by a transaction keeping track of the server resources and the states of the VMs allocated to the server. In wireless sensor networks, there can be a long-range communication between pairs of agents in different locations. In chemistry, compounds can be thought of as transactions between the individual molecules which are able to jointly take part in further reactions.

In this chapter, we describe how to extend our framework to capture such transactional interaction pattern in a concise way. We use the client–server model as a simple example. Instead of representing the client and server agents individually, we can include a new auxiliary population which tracks the number of pairs of clients and servers in the "$(Client\_wait, Server\_get)$" ongoing transaction. This model can be still represented as a PCTMC. It turns out that in this case, the resulting PCTMC stays the same – the populations $C_w(t)$ and $S_g(t)$ are identical and only get replaced by the new population, say $C_w S_g(t)$. However, the situation changes if we introduce a new class of client agents. Suppose that instead of one type of clients, there are two different classes, $Client_1$ and $Client_2$, with identical state transitions as in $Client$, only with each state with a subscript 1 or 2 respectively. Additionally, the $Client_2$ class clients have a ten times slower

data transition rate. The system equation becomes:

$$\mathbf{Clients}\{Client_1[n_1] \parallel Client_2[n_2]\} \underset{\{request,data\}}{\bowtie} \mathbf{Servers}\{Server[n_S]\}$$

We compare two different interpretations of the model – a standard GPEPA interpretation where the request and data stages are independent and any server can serve data to any client and a transactional interpretation where the model keeps track of the number of pairs of a server agent in the $Server\_get$ state and one of the two client agent in the state $Client\_wait_1$ or $Client\_wait_2$ respectively.

In either case, we can apply the ODE analysis to the underlying PCTMC and access performance metrics such as passage times and rewards mentioned in previous chapters. The analysis reveals significant quantitative differences between the two models. Figure 7.1 shows the CDF of the passage time of a single client from $Client_1$ class performing its first $think$ action. In the transaction version of the model, the probabilities are lower throughout the whole time. This is caused by the slower clients blocking the servers after the request stage, where in the original model the servers in the $Server\_get$ stage would be equally available to both client classes.

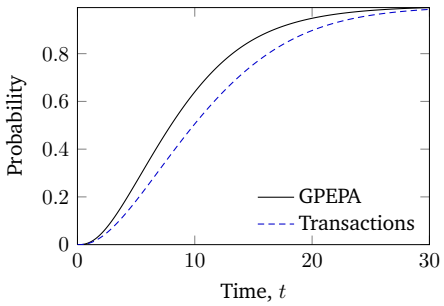


Figure 7.1: Client passage-time CDF in the two-class client–server model; comparison between the GPEPA version and a PCTMC model with transactions.

The main contributions of this chapter are the following:

GPEPAc – GPEPA with channels We introduce a lightweight extension to the GPEPA stochastic process algebra that allows us to elegantly express transaction cooperation which is orthogonal to the group structure of the model. In general, it is possible to represent transactions via auxiliary populations in the PCTMC model. However, often there can be a large number of different combinations of agents forming a transaction and manual specification of such models can become cumbersome.

Computing cluster case study We present a larger case study where we use the channel extension of GPEPA to model a large-scale computing cluster where servers are capable of concurrently running a number of jobs. We allow a number of server and job classes each with different performance characteristics. We look at the performance–energy trade-off from Section 5.5, considering multiple SLAs corresponding to each job class.

Our approach is in contrast to the approach by Hayden and Bradley [102], who present *Shared Transaction Markov Chains*, a low level framework that can be potentially adapted to different formalisms.

7.2 GPEPAc – GPEPA with channels

In this section, we introduce an extension of the GPEPA process algebra that allows a flexible specification of transactional cooperation. We add a lightweight syntactical layer to GPEPA and re-use the original GPEPA semantics when defining the underlying PCTMC. We extend the description of PEPA agents with *sockets*. Two sockets belonging to multiple agents, originating in different groups in the model, can be joined by a *channel*. The channels are created after standard GPEPA action interaction, and are building blocks of transactions. Agents can perform actions on sockets, which restrict the cooperation to only the agents sharing the same channel. Each agent can maintain a number of sockets joined to a number of channels. Each transaction can be therefore represented as a connected graph of GPEPA group–agent pairs, joined via named sockets. The resulting PCTMC model will include a population for each such graph. The Grouped Model syntax of GPEPA is extended with annotations which specify the transaction each agent belongs to.

7.2.1 Extended syntax of PEPA agents

The extension of PEPA agent syntax consists of two parts – *agents with sockets* and *actions with socket instructions*. Each agent is allowed to maintain a set of sockets $\{\underline{s}_1, \ldots, \underline{s}_k\}$ from a set of socket labels $\mathcal{S}$:

$$P^{\{\underline{s}_1,\cdots,\underline{s}_k\}} \qquad\qquad \underline{s}_i \in \mathcal{S}$$

We usually omit the braces above and write a set of sockets in uppercase letters, $P^{\underline{S}}$ for the agent above. For example, the client agent could maintain a connection to the server serving the requests via a socket $\underline{s}$, $Client\_wait^{\underline{s}}$, and a server could remember the respective client with a socket $\underline{c}$, $Server\_get^{\underline{c}}$. In a different model, a server could keep track of two served clients, $Server\_get^{\underline{c}_1,\underline{c}_2}$.

Agents can execute actions annotated with *socket instructions* in order to establish new persistent connections or to restrict the communication onto a connection maintained by the socket. We allow four instructions: send on, receive on, init, forward and get. Each can annotate an action a with a socket $\underline{s}$ (or a list of sockets in case of the send on instruction), e.g. $(a \,\mathsf{send\ on}\, \underline{s}).P$. In the definition of an agent $P^{\underline{S}}$, $\underline{S} = \{\underline{s}_1, \ldots, \underline{s}_k\}$ we allow the following action prefixes (the description of their behaviour is only informal and is defined in GPEPAc semantics in Section 7.3):

The init **instruction** An action a can be annotated with $\mathsf{init}\ \underline{z}$, where $\underline{z}$ is a new socket not present in $\underline{S}$. Executing the action results in creation of a new socket and a corresponding channel which is sent to agents cooperating on the action a. The other agents receive the new channel via a corresponding get instruction, see below.

$$(a \,\mathsf{init}\, \underline{z}).Q^{\underline{U}} \qquad\qquad \underline{z} \notin \underline{S}, \underline{U} \subseteq \underline{S} \cup \{\underline{z}\}$$

The forward **instruction** An action a can be annotated with $\mathsf{forward}\ \underline{s}$, $\underline{s} \in \underline{S}$. This works similarly to the init instruction above, but instead of creating a new socket–channel pair, an existing channel is forwarded to the receiving agents within a cooperation.

$(a \text{ forward } \underline{s}).Q^{\underline{U}}$ $\quad\quad$ $\underline{s} \in \underline{S}, \underline{U} \subseteq \underline{S}$

The get **instruction** An action a can be annotated with get $\underline{z}$, where $\underline{z}$ is either an existing or a new socket. The agent accepts a new channel after a cooperation on a and binds it to the socket $\underline{z}$.

$(a \text{ get } \underline{z}).Q^{\underline{U}}$ $\quad\quad$ $\underline{U} \subseteq \underline{S} \cup \{z\}$

The send on **and** receive on **instructions** An action a can be annotated with send on $\underline{S'}$ where $\underline{S'} \subseteq \underline{S}$ are existing sockets or with receive on $\underline{s}$ where $\underline{s} \in \underline{S}$ is an existing socket. Cooperation on such action a is then restricted to one agent with the send on instruction and a number of agents with receive on instruction, such that they share the channels attached to sockets in $\underline{S'}$. We assume that the actions with receive on and send on instructions are distinct from all the remaining actions in the system.

$(a \text{ send on } \underline{S}).Q^{\underline{U}}$ $\quad\quad$ $\underline{S'} \subseteq \underline{S}, \underline{U} \subseteq \underline{S}$
$(a \text{ receive on } \underline{s}).Q^{\underline{U}}$ $\quad\quad$ $\underline{s} \in \underline{S}, \underline{U} \subseteq \underline{S}$

For example, a server can establish a new connection with a specific client after the request action with the init instruction and the corresponding client accepts the connection by creating a new socket with the get instruction:

$$Server = (request \text{ init } \underline{c}, r_{request}).Server\_get^{\underline{c}}$$
$$Client = (request \text{ get } \underline{s}, r_{request}).Client\_wait^{\underline{s}}$$

In this way, it is possible to restrict the cooperation on the subsequent $data$ action to only the pairs of client and server agents whose sockets are joined by the same channel (this is defined in the semantics below). We can re-define the $Client\_wait$ and $Server\_get$ states:

$$Client\_wait^{\underline{s}} = (data \text{ receive on } \underline{s}, r_{data}).Client\_think$$
$$Server\_get^{\underline{c}} = (data \text{ send on } \underline{c}, r_{data}).Server$$

> To demonstrate all the features of GPEPAc, we use a modified version of the client–server model where each server can serve two clients simultaneously. The persistent session with each client is kept via a corresponding socket combination. To demonstrate a communication over multiple sockets, a server can break when serving two clients (e.g. by running out of memory), notifying both of them at the same time. The server agent definition is:
>
> $$\begin{aligned}
> Server &= (request \text{ init } \underline{c}_1, r_{request}).Server_get^{\underline{c}_1} + (break, r_{break}).Server_broken \\
> Server_get^{\underline{c}_1} &= (request \text{ init } \underline{c}_2, r_{request}).Server_get^{\underline{c}_1,\underline{c}_2} + (data \text{ send on } \underline{c}_1, r_{data}).Server \\
> Server_get^{\underline{c}_2} &= (request \text{ init } \underline{c}_1, r_{request}).Server_get^{\underline{c}_1,\underline{c}_2} + (data \text{ send on } \underline{c}_2, r_{data}).Server \\
> Server_get^{\underline{c}_1,\underline{c}_2} &= (data \text{ send on } \underline{c}_1, r_{data,2}).Server_get^{\underline{c}_2} \\
> &\quad + (data \text{ send on } \underline{c}_2, r_{data,2}).Server_get^{\underline{c}_1} \\
> &\quad + (break \text{ send on } \underline{c}_1, \underline{c}_2, r_{break}).Server_broken
> \end{aligned}$$

$$Server\_broken = (reset, r_{reset}).Server$$

The corresponding client agent definition is ($i \in \{1, 2\}$):

$$\begin{aligned} Client_i &= (request \text{ get } \underline{s}, r_{request}).Client\_wait_i^{\underline{s}} \\ Client\_wait_i^{\underline{s}} &= (data \text{ receive on } \underline{s}, r_{data}).Client\_think_i + (break \text{ receive on } \underline{s}, r_{break}).Client_i \\ Client\_think_i &= (think, r_{think}).Client_i \end{aligned}$$

We will interleave the definitions below with demonstrations on this example and will highlight the corresponding passages in the same style as this text.

7.2.2 Extended syntax of GPEPA models

In addition to the grouped model structure of GPEPA, models in GPEPAc maintain information about the ongoing *transactions*.

Channels and transactions

The building blocks of transactions are channels. Each agent with sockets can be given a *channel assignment* $\Gamma\colon \mathcal{S} \to \mathcal{C}h$, where $\mathcal{C}h$ is the set of all channels. We write

$$P^{\underline{S}}(\Gamma).$$

We use lowercase letters of the Greek alphabet to denote channels. For example, a client agent can be given an assignment $\Gamma = \{\underline{s} \mapsto \alpha\}$ where the socket $\underline{s}$ is attached to a channel α.

We consider multisets of group–agents pairs with channel assignments, written as

$$\tau = \left[\!\!\left[G_1 : P_1^{\underline{S}_1}(\Gamma_1), \ldots, G_l : P_l^{\underline{S}_l}(\Gamma_l) \right]\!\!\right]$$

where G_i are group labels from the GPEPAc model (see below).

Each such multiset can be represented as a bi-partite graph where agent nodes are joined via named sockets to channels. We call such a multiset a *transaction* if the corresponding graph is connected.

In the client–server model, a possible transaction would be a server connected to a client from the first class on socket $\underline{c}_1$:

$$\tau_1^1 = \left[\!\!\left[Client\_wait_1^{\underline{s}}(\{\underline{s} \mapsto \alpha\}), Server\_get^{\underline{c}_1}(\{\underline{c}_1 \mapsto \alpha\}) \right]\!\!\right]$$

We consider transactions modulo channel names unless stated otherwise. For example, the transaction τ_1^1 above would stay identical if we replace the channel α with a channel β.

We allow standard multiset operators such as union and difference. We also use *substitution*. Take a transaction τ and a set of group agent pairs $G_i : P_i^{\underline{S}_i}(\Gamma_i)$ where each agent changes state to $Q_i^{\underline{Z}_i}(\Gamma_i')$. Then τ after the corresponding substitution is:

$$\tau[G_1 : P_1^{\underline{S}_1}(\Gamma_1) \mapsto Q_1^{\underline{Z}_1}(\Gamma_1'), \ldots, G_l : P_l^{\underline{S}_l}(\Gamma_l) \mapsto Q_l^{\underline{Z}_l}(\Gamma_l')]$$

$$:= \tau \setminus [\![G_1 : P_1^{S_1}(\Gamma_1), \ldots, G_l : P_l^{S_l}(\Gamma_l)]\!] \cup [\![G_1 : Q_1^{Z_1}(\Gamma'_1), \ldots, G_l : Q_l^{Z_l}(\Gamma'_l)]\!]$$

The union of two transactions or a substitution on a single transaction does not necessarily result in a multiset that is a transaction. We define a function $Split(\tau)$ which splits a multiset τ of agents with channel assignments into a multiset of transactions which correspond to the maximal connected components in τ.

> As defined in the semantics of GPEPAc below, the transaction τ_1^1 can split after a data transition, resulting in two transactions with single client and server agents respectively:
>
> $$Split\left(\tau_1^1[\ \mathbf{Clients} : Client\_wait_1^s(\{\underline{s} \mapsto \alpha\}) \mapsto Client\_think_1,\right.$$
> $$\left.\mathbf{Servers} : Server\_get^{c_1}(\{\underline{c}_1 \mapsto \alpha\}) \mapsto Server]\right) = [\![[\![Client\_think_1]\!], [\![Server]\!]]\!]$$

Models with transactions

Models in GPEPAc consist of the same group structure as in GPEPA, but additionally contain transaction annotations of agents in the following syntax:

$$G := G \underset{L}{\bowtie} G \ \mid\ G \parallel G \ \mid\ \mathbf{Y}\{\tau \ni P(\Gamma) \parallel \cdots \parallel \tau \ni P(\Gamma)\}$$

The notation $\tau \ni P(\Gamma)$ corresponds to an agent P with an assignment Γ that is part of a transaction τ. If τ is a transaction consisting of a single agent, we normally omit the annotation. Usually, the initial state of the model (given by the system equation), does not contain any ongoing transaction (of more than one agent). Therefore, a user specifying a GPEPAc model does not have to be concerned with explicitly writing out any transactions. We write $\tau \in M$ if there is a group G in a model M containing an agent $\tau \ni P$.

> The GPEPAc system equation of the client–server model is
>
> $$\mathbf{Clients}\{Client_1[n_1] \parallel Client_2[n_2]\} \underset{request}{\bowtie} \mathbf{Servers}\{Server[n_S]\}$$
>
> After one request of a client to a server, the model changes to
>
> $$\mathbf{Clients}\{Client_1[n_1 - 1] \parallel Client_2[n_2] \parallel \tau_1^1 \ni Client\_wait_1^s(\{\underline{s} \mapsto \alpha\})\}$$
> $$\underset{request}{\bowtie} \mathbf{Servers}\{Server[n_S - 1] \parallel \tau_1^1 \ni Server\_get^{c_1}(\{\underline{c}_1 \mapsto \alpha\})\}$$
>
> We define this behaviour formally in the PCTMC semantics of GPEPAc below.

7.3 PCTMC semantics of GPEPAc

In PCTMCs derived from GPEPA models, each population corresponds to a group–agent pair. In GPEPAc, populations correspond to ongoing transactions between a multiset of group–agent pairs. In GPEPA, the global set of all group–agent pairs can be generated by independently considering all derivative states for each agent. In GPEPAc, this is no longer possible, as new

combinations of agents in transactions can arise after transitions in the system. Similarly, in GPEPA the possible transition classes can be directly enumerated from the possible transitions of individual agents. In GPEPAc, the complete set of transition classes can be derived only once all the possible transactions are known.

We define an iterative algorithm which simultaneously computes the set of all transactions and transition classes in a GPEPAc model. There are two types of transition classes in GPEPAc – *internal* transitions corresponding to communication on channels inside a transaction and *external* transitions corresponding to communication between agents from different transactions. This agent communication is identical to the action cooperation in GPEPA. Therefore the semantics of external transition classes is induced by transition classes in an auxiliary GPEPA model. The semantics of internal transition classes is similar to transitions of single agents and is defined below.

7.3.1 Algorithm to compute the set of all transactions and transition classes

The algorithm explores the possible populations and transition classes in a GPEPAc model in an iterative fashion. Similar to simulation, at each step the algorithm enumerates all the possible transition classes for current transactions in the model. The rates of the transitions are ignored and the algorithm selects a transition class which introduces a new transaction into the model. Additionally, the population sizes are considered to be large enough to always allow each possible transition class (we will denote this by writing ∞ for the number of agent replications in the system equation). The algorithm terminates when all possible transition classes only change populations of existing transactions in the system.

In each iteration, the algorithm collects all the possible internal and external transition classes. The set $\mathbb{E}$ stores information about all the possible external transition classes. It consists of triples

$$\left(\left\{ G : {}^{\tau_i,\Gamma_i}P_i^{\underline{S}_i} \xrightarrow{(a_i,r_i)} {}^{\tau_i,\Gamma}Q_i^{\underline{Z}_i} \right\}_{i=1}^{l}, In, Out \right)$$

The first element is a set of agent transitions in an auxiliary GPEPA model that we define shortly. The second and third elements are incoming and outgoing transactions that will define the change vector of the resulting transition class. After the last iteration, the algorithm can use each triple to derive a complete transition class, including the rate function.

The set $\mathbb{I}$ stores all internal transition classes. Their rates do not depend on the rest of the model and so they can be directly stored as $([\![\tau_{in}]\!], Out, r)$ where $[\![\tau_{in}]\!]$ is the only incoming transaction and Out is a multiset of outgoing transactions and $r \in \mathbb{R}_+$ is a rate constant. This represents a transition class

$$\tau_{in} \to \sum_{\tau \in Out} \tau \qquad \text{at } \boldsymbol{X}_{\tau_{in}} \cdot r$$

The algorithm starts with a GPEPAc system equation M. As a first step, each agent is replaced with infinitely many copies of itself. Let this model be $M^{(0)}$. At each step k the algorithm will maintain a current version of the model, $M^{(k)}$ and the sets $\mathbb{E}$ and $\mathbb{I}$.

At each step, the algorithm examines all the possible internal and external transitions classes in $M^{(k)}$ and stores information about them in the sets $\mathbb{E}$ and $\mathbb{I}$. Each transition class results in a change of a number of transactions, reflected in a potential candidate for the model $M^{(k+1)}$. Because of the "infinite" populations in $M^{(k)}$, this new model contains all transactions encountered so far. If it additionally contains a new unseen transaction, the algorithm continues with the step $k+1$. Otherwise this process is repeated for the remaining transition classes. If none introduce a new transaction, the algorithm terminates.

A high-level overview of the algorithm can be seen in Algorithm 1. The overall complexity of the algorithm depends on the total number of different transactions in the model. Certain model structures could potentially give rise to infinitely many distinct transactions. For example if an agent A in group G could establish a connection with up to two different agents B from group H and each agent B could establish a connection with up to two different agents A, there could be transactions of the form $[\![G\colon A(\Gamma_1), H\colon B(\Gamma_2), \ldots, H\colon B(\Gamma_{k-1}), G\colon A(\Gamma_k)]\!]$ of an arbitrary size. In this chapter, we assume that the number of different transactions is finite and so that the algorithm terminates.

Algorithm 1: Computing all possible transactions in a GPEPAc model

input : M – a GPEPAc model
output: $M^{(\infty)}$ – a GPEPAc model
$\mathbb{I}$ – set of internal transition classes
$\mathbb{E}$ – set of specifications of external transition classes

1 **begin**
2 Set $M^{(0)} = M$, replace each population by ∞
3 **while** *possible to find* $M^{(k+1)} \neq M^{(k)}$ **do**
4 Generate GPEPA model $M^{(k)}_{GPEPA}$ ◁ **External transitions** 7.3.1.1
5 **foreach** *transition class* c *of* $M^{(k)}_{GPEPA}$ **do**
6 Translate the effect of c on $M^{(k)}$:
7 join and split transactions according to the channel semantics
8 resulting model $M^{(k+1)}$, replace each population by ∞
9 Collect the sets of incoming and outgoing transactions in $M^{(k)}$, In, Out, store (c, In, Out) in $\mathbb{E}$
10 **if** $M^{(k+1)} \neq M^{(k)}$ **then**
11 $k \leftarrow k+1$, **continue at 4**
12 **foreach** *transaction* τ *in* $M^{(k)}$ ◁ **Internal transitions** 7.3.1.2
13 **do**
14 **foreach** *possible socket communication* c *in* τ **do**
15 Translate the effect of c on $M^{(k)}$:
16 $In = [\![\tau]\!]$, Out is $Split(\tau$ after $c)$
17 resulting model $M^{(k+1)}$, replace each population by ∞
18 Store $(In, Out, \tau \cdot \min(r_i))$ in $\mathbb{I}$
19 **if** $M^{(k+1)} \neq M^{(k)}$ **then**
20 $k \leftarrow k+1$, **continue at 4**
21 Set $M^{(\infty)} = M^{(k)}$

7.3.1.1 External transitions

To enumerate the possible external transition classes, we define an auxiliary GPEPA model $M^{(k)}_{GPEPA}$ as follows: replace each $\tau \ni P^{\underline{S}}(\Gamma)$ in a group G with an agent ${}^{\tau,\Gamma}P^{\underline{S}}$ that is defined exactly as $P^{\underline{S}}$, but with each derivative state also prefixed with "$^{\tau,\Gamma}$". Ignore any socket annotations and actions with send on and receive on instructions. The PCTMC semantics of $M^{(k)}_{GPEPA}$ gives transition classes consisting of transitions of individual agents and a rate function defined on vectors of populations of group-agent pairs in $M^{(k)}_{GPEPA}$:

$$\left(\underbrace{\left\{ G : {}^{\tau_i,\Gamma_i}P_i^{\underline{S}_i} \xrightarrow{(a_i,r_i)} {}^{\tau_i,\Gamma_i}Q_i^{\underline{Z}_i} \right\}_{i=1}^{l}}_{T}, r(\boldsymbol{X}) \right) \tag{7.1}$$

Each of these transition classes defines an external transition class in GPEPAc. The effect of this transition class additionally takes into account the socket instructions to produce the multiset of outgoing transactions. The multiset of incoming transactions is given by

$$In = [\![\tau_i]\!]_{i=1}^{l}$$

There are two possible cases for the multiset of outgoing transactions, which depend on the interaction structure.

GPEPA interactions not affecting sockets All the socket sets $\underline{S}_i$ and $\underline{Z}_i$ are empty. This implies that all the involved transactions contain only a single agent. In the GPEPAc model $M^{(k)}$, the corresponding transition class simply changes transactions $\tau_i = [\![G_i : P_i]\!]$ into $\tau_i' = [\![G_i : Q_i]\!]$. Therefore set

$$Out = [\![[\![G_i : Q_i]\!]]\!]_{i=1}^{l}$$

We extend $\mathbb{E}$ with the triple (if $\mathbb{E}$ does not contain it already):

$$\left(T, [\![G_i : P_i]\!]_{i=1}^{l}, [\![G_i : Q_i]\!]_{i=1}^{l}\right)$$

The resulting candidate for the model $M^{(k+1)}$ after such transition then additionally contains agents Q_i in group G_i for each $1 \le i \le l$. We denote this as

$$M^{(k+1)} = M^{(k)} + \{G_i : \tau_i' \ni Q_i[\infty]\}_{i=1}^{l}$$

If $M^{(k+1)}$ is different from $M^{(k)}$, we accept it and continue at the next step.

> In the client–server model, we can have
>
> $$M^{(0)} = \mathbf{Clients}\{\mathit{Client}_1[\infty] \parallel \mathit{Client}_2[\infty]\} \underset{request}{\bowtie} \mathbf{Servers}\{\mathit{Server}[\infty]\}$$

(as mentioned before, we leave out the transaction annotations for single agent transactions). One possible transition class corresponds to a server failure, given by the following tuple

$$\left(\left\{\mathbf{Servers} : \mathit{Server} \xrightarrow{(break,\, r_{break})} \mathit{Server\_broken}\right\}, S \cdot r_{break}\right)$$

The incoming and outgoing transaction multisets are:

$$\begin{aligned} In &= [\![[\![\mathit{Server}]\!]]\!] \\ Out &= [\![[\![\mathit{Server\_broken}]\!]]\!] \end{aligned}$$

Applying this transition to $M^{(0)}$ we get

$$\begin{aligned} M^{(1)} = \mathbf{Clients}\{&\mathit{Client}_1[\infty] \parallel \mathit{Client}_2[\infty]\} \\ &\underset{request}{\bowtie} \mathbf{Servers}\{\mathit{Server}[\infty] \parallel \mathit{Server\_broken}[\infty]\}. \end{aligned}$$

Applying the same transition to $M^{(1)}$ in the next step would result in the same model and therefore a different transition class has to be found in order to proceed to step 2.

Transitions affecting socket connections One of the socket list $\underline{S}_i$ or $\underline{Z}_i$ is non-empty. This means that this transition class potentially results in a creation of new transactions – either by splitting some of the transactions in In due to agents closing sockets or by merging some of the transactions via the init–get interaction. There are three possibilities for the type of action annotations in a_i:

- All $a_i = a$ for $1 \leq i \leq l$ and a an action label. No new sockets are created in this transition – either the incoming transactions from In undergo a change in the state of the cooperating agent or undergo a structural change in their channel graph by closing sockets, possibly resulting in transactions being split into a number of smaller transactions. The resulting outgoing transactions are

$$Out = \bigcup_{i=1}^{l} Split\left(\tau_i\left[G_i : P_i^{\underline{S}_i}(\Gamma_i) \mapsto G_i : Q_i^{\underline{Z}_i}(\Gamma_i')\right]\right)$$

  where Γ_i' is a restriction of Γ_i onto the socket list $\underline{Z}_i \subseteq \underline{S}_i$.

- One a_i, say a_1 is of the form $a\,\mathsf{init}\,\underline{u}_1$ and the remaining a_j of the form $a\,\mathsf{get}\,\underline{u}_j$ for $2 \leq j \leq l$. In this case new sockets are created and are joined by a newly created channel. Let α be a new channel not present in any of the incoming transactions τ_i. The outgoing transactions are

$$Out = Split\left(\bigcup_{i=1}^{l} \tau_i\left[G_i : P_i^{\underline{S}_i}(\Gamma_i) \mapsto Q_i^{\underline{Z}_i}(\Gamma_i')\right]\right)$$

where

$$\Gamma'_i(\underline{v}) = \begin{cases} \Gamma_i(\underline{v}) & \text{for } \underline{v} \in \underline{Z}_i,\ \underline{v} \neq \underline{u}_i \\ \alpha & \text{for } \underline{v} = \underline{u}_i \end{cases}$$

- One a_i, say a_1 is of the form a forward $\underline{u}_1$ and the rest of the form a get $\underline{u}_j$ for $2 \leq j \leq 1$. This is similar to the case above, with $\alpha = \Gamma_1(\underline{u}_1)$ instead of a new channel.

In both cases, we extend $\mathbb{E}$ with the corresponding triple (T, In, Out). The candidate for $M^{(k+1)}$ contains the agents which actively changed state in the transition class. Additionally, the remaining agents from each transaction in In change their transaction annotations to a corresponding transaction in Out. The resulting model is:

$$M^{(k)} + \{G_i : \tau \ni Q(\Gamma) \mid G : Q(\Gamma) \in \tau, \tau \in Out\} \tag{7.2}$$

In the client–server model, an interaction between a client and a server is represented by the transition class in $M^{(1)}_{GPEPA}$

$$\left(\left\{ \begin{array}{l} \mathbf{Clients} : Client_1 \xrightarrow{(request\ \mathsf{get}\ \underline{s},\, r_{request})} Client\_wait_1^{\underline{s}}, \\ \mathbf{Servers} : Server \xrightarrow{(request\ \mathsf{init}\ \underline{s},\, r_{request})} Server\_get^{\underline{c}_1} \end{array} \right\}, \right.$$
$$\left. \frac{C_1}{C_1 + C_2} \min(S, C_1 + C_2) \cdot r_{request} \right)$$

The incoming and outgoing transactions are

$$In = [\![[\![\mathbf{Clients} : Client_1]\!], [\![\mathbf{Servers} : Server]\!]]\!]$$
$$Out = \left[\!\left[\tau_1^1\right]\!\right]$$

and the next model

$$M^{(2)} = \mathbf{Clients}\{Client_1[\infty] \parallel Client_2[\infty] \parallel \tau_1^1 \ni Client\_wait_1^{\underline{s}}(\{\underline{s} \mapsto \alpha\})[\infty]\}$$
$$\underset{request}{\bowtie} \mathbf{Servers}\{Server[\infty] \parallel \tau_1^1 \ni Server\_get^{\underline{c}_1}(\{\underline{c}_1 \mapsto \alpha\})[\infty]$$
$$\parallel Server\_broken[\infty]\}$$

In $M^{(2)}$, a server can accept another client, say from the class $Client_1$. This is determined by a transition class in $M^{(2)}_{GPEPA}$:

$$\left(\left\{ \begin{array}{l} \mathbf{Clients} : Client_1 \xrightarrow{(request\ \mathsf{get}\ \underline{s},\, r_{request})} Client\_wait_1^{\underline{c}}, \\ \mathbf{Servers} : {}^{\tau_1^1,\Gamma} Server\_get^{\underline{c}_1} \xrightarrow{(request\ \mathsf{init}\ \underline{c}_2,\, r_{request})} Server\_get^{\underline{c}_1,\underline{c}_2} \end{array} \right\}, \right.$$

$$\left.\frac{C_1}{C_1+C_2}\frac{\tau_1^1,\Gamma\, S_g^{\underline{c}_1}}{S+{}^{\tau_1^1,\Gamma}S_g^{\underline{c}_1}}\min(S+{}^{\tau_1^1,\Gamma}S_g^{\underline{c}_1},C_1+C_2)\cdot r_{request}\right)$$

The incoming multiset of transaction is

$$In=\left[\!\left[\tau_1^1,[\![\mathbf{Clients}:\mathit{Client}_1]\!]\right]\!\right].$$

The multiset of outgoing transactions consists of a single transaction

$$\begin{aligned}\tau_{12}^{11}=[\![&\mathbf{Clients}:\mathit{Client\_wait}_1^{\underline{s}}(\{\underline{s}\mapsto\alpha\}),\mathbf{Clients}:\mathit{Client\_wait}_1^{\underline{s}}(\{\underline{s}\mapsto\beta\}),\\&\mathbf{Servers}:\mathit{Server\_get}^{\underline{c}_1,\underline{c}_2}(\{\underline{c}_1\mapsto\alpha,\underline{c}_2\mapsto\beta\})]\!]\end{aligned}$$

The resulting model $M^{(3)}$ after the transition is

$$\begin{aligned}M^{(3)}=\mathbf{Clients}\{&\mathit{Client}_1[\infty]\parallel\mathit{Client}_2[\infty]\parallel\tau_1^1\ni\mathit{Client\_wait}_1^{\underline{s}}(\{\underline{s}\mapsto\alpha\})[\infty]\\&\parallel\tau_{12}^{11}\ni\mathit{Client\_wait}_1^{\underline{s}}(\{\underline{s}\mapsto\alpha\})[\infty]\parallel\tau_{12}^{11}\ni\mathit{Client\_wait}_1^{\underline{s}}(\{\underline{s}\mapsto\beta\})[\infty]\}\\&\underset{request}{\bowtie}\\\mathbf{Servers}\{&\mathit{Server}[\infty]\parallel\tau_1^1\ni\mathit{Server\_get}^{\underline{c}_1}(\{\underline{c}_1\mapsto\alpha\})[\infty]\parallel\mathit{Server\_broken}[\infty]\\&\parallel\tau_{12}^{11}\ni\mathit{Server\_get}^{\underline{c}_1,\underline{c}_2}(\{\underline{c}_1\mapsto\alpha,\underline{c}_2\mapsto\beta\})[\infty]\}\end{aligned}$$

7.3.1.2 Internal transitions

Each transaction in a GPEPAc model allows its agents to communicate on the established channel connections. Each such communication consists of one agent executing an action with a send on instruction and a number of agents executing the same action with receive on instructions. Each receive on instruction has to act on a socket that shares a channel with the sending agent.

At each step, consider every transaction τ in $M^{(k)}$. Take every agent $G:P^{\underline{S}}(\Gamma)$ in τ that is capable of executing an action a with the instruction send on $\underline{u}_1,\ldots,\underline{u}_l$, that is there exists a transition

$$P^{\underline{S}}\xrightarrow{(a\text{ send on }\underline{u}_1,\ldots,\underline{u}_l,r)}Q^{\underline{Z}}$$

Find all agents capable of executing the same action on one of the channels connected to the sockets $\underline{u}_j$. That is agents $G_i:P_i^{\underline{S}_i}(\Gamma_i)$ in τ capable of executing the action a receive on $\underline{v}_i$, $1\le i\le k$,

$$P_i^{\underline{S}_i}\xrightarrow{(a\text{ receive on }\underline{v}_i,r_i)}Q_i^{\underline{Z}_i}$$

such that $\Gamma_i(\underline{v}_i)\in\{\Gamma(\underline{u}_j)\mid 1\le j\le l\}$. We require that there is at least one such agent, otherwise this transition is not allowed. We could define different versions of internal transitions which impose different restrictions on the interaction, for example blocking the transition until there is a corresponding agent for each channel attached to $\underline{u}_i$.

All the agents P and P_i, $1\le i\le k$, evolve simultaneously and so the resulting outgoing transactions can be obtained by substituting the agents for the right-hand sides of the above

transitions:

$$Out = Split\left(\tau\left[G : P^{\underline{S}}(\Gamma) \mapsto Q^{\underline{Z}}(\Gamma'), G_i : P_i^{\underline{S}_i}(\Gamma_i) \mapsto Q_i^{\underline{Z}_i}(\Gamma'_i), 1 \leq i \leq k\right]\right)$$

where Γ' and Γ'_i are restrictions of Γ and Γ_i to $\underline{Z}$ and $\underline{Z}_i$ respectively. The incoming multiset of transactions consists only of the transaction τ, $In = [\![\tau]\!]$. We define the rate of this transition as the minimum of all the involved rates, $r_{\min} = \min(r, r_1, \ldots, r_k)$. We can include the triple $(In, Out, r_{\min})$ in the set $\mathbb{I}$. Similarly to the case of external transitions, the candidate for the next model contains all the newly transformed agents, Equation 7.2. If this model contains new transactions, the algorithm continues at the next step $k + 1$.

For example, model $M^{(3)}$ contains the transaction τ_{12}^{11}. The server agent from this transaction, $\mathbf{Servers} : Server\_get^{\underline{c}_1,\underline{c}_2}(\{\underline{c}_1 \mapsto \alpha, \underline{c}_2 \mapsto \beta\})$ is capable of sending data to say the client listening to the channel α on socket $\underline{c}_1$,

$$Server\_get^{\underline{c}_1,\underline{c}_2} \xrightarrow{(data\ \mathsf{send\ on}\ \underline{c}_1, r_{data})} Server\_get^{\underline{c}_2}$$

The client listening to the channel α is capable of executing the corresponding receiving action

$$Client\_wait_1^{\underline{s}} \xrightarrow{(data\ \mathsf{receive\ on}\ \underline{s}, r_{data,1})} Client\_think_1$$

and so the resulting multiset of outgoing transactions is

$$Out = \left[\!\!\left[\tau_2^1, [\![\mathbf{Clients} : Client\_think_1]\!]\right]\!\!\right]$$

where τ_2^1 is a transaction between a server in the $Server\_get$ state with a client of the first class connected to the socket $\underline{c}_2$. The resulting model $M^{(4)}$ is

$$\begin{aligned} M^{(4)} = \mathbf{Clients}\{&Client_1[\infty] \parallel Client_2[\infty] \parallel \tau_1^1 \ni Client\_wait_1^{\underline{s}}(\{\underline{s} \mapsto \alpha\})[\infty] \\ &\parallel \tau_{12}^{11} \ni Client\_wait_1^{\underline{s}}(\{\underline{s} \mapsto \alpha\})[\infty] \parallel \tau_{12}^{11} \ni Client\_wait_1^{\underline{s}}(\{\underline{s} \mapsto \beta\})[\infty] \\ &\parallel \tau_2^1 \ni Client\_wait_1^{\underline{s}}(\{\underline{s} \mapsto \beta\})[\infty] \parallel Client\_think_1[\infty]\} \\ &\underset{request}{\bowtie} \\ \mathbf{Servers}\{&Server[\infty] \parallel \tau_1^1 \ni Server\_get^{\underline{c}_1}(\{\underline{c}_1 \mapsto \alpha\})[\infty] \parallel Server\_broken[\infty] \\ &\parallel \tau_{12}^{11} \ni Server\_get^{\underline{c}_1,\underline{c}_2}(\{\underline{c}_1 \mapsto \alpha, \underline{c}_2 \mapsto \beta\})[\infty] \\ &\parallel \tau_2^1 \ni Server\_get^{\underline{c}_2}(\{\underline{c}_2 \mapsto \beta\})[\infty]\} \end{aligned}$$

Another possible internal transition in the transaction τ_{12}^{11} corresponds to the $break$ action announced by the server to both clients. We have

$$Server\_get^{\underline{c}_1,\underline{c}_2} \xrightarrow{(break\ \mathsf{send\ on}\ \underline{c}_1,\underline{c}_2, r_{break})} Server\_broken$$

and

$$\mathit{Client\_wait}_1^{\underline{s}} \xrightarrow{(\mathit{break}\ \mathsf{receive\ on}\ \underline{s}, r_{\mathit{break}})} \mathit{Client}_1$$

The multiset of outgoing transactions now contains two copies of the single client agent and a single broken server:

$$\mathit{Out} = [\![2 \times [\![\mathbf{Clients} : \mathit{Client}_1]\!], [\![\mathbf{Servers} : \mathit{Server\_broken}]\!]]\!] .$$

The model $M^{(5)}$ is obtained by adding the agent *Server_broken* to the group **Servers** in $M^{(4)}$.

7.3.2 Computing the final PCTMC

We use the previous algorithm to obtain the final PCTMC of a GPEPAc model M. Algorithm 2 gives an overview. We show how to derive the population vector $\boldsymbol{X}(t)$, set of transition classes $\mathcal{C}$ and the initial populations $\boldsymbol{X_0}$.

Algorithm 2: Generating a PCTMC from a GPEPAc model

input : M – a GPEPAc model
output: PCTMC $\boldsymbol{X}(t), \mathcal{C}, \boldsymbol{X_0}$ corresponding to M

begin
- Use Algorithm 1 to compute $M^{(\infty)}$, $\mathbb{I}$, $\mathbb{E}$ from M.
- Populations labels correspond to transactions in $M^{(\infty)}$
- Set the initial populations as $X_{0,\tau} = \#_\tau(M)$ for $\tau \in M^{(\infty)}$
- **foreach** *transition specification* $([\![\tau_{in}]\!], \mathit{Out}, r)$ *in* $\mathbb{I}$ **do** ◁ **Internal transitions**
  - Add the transition class below to $\mathcal{C}$:
    $$\tau_{in} \to \textstyle\sum_{\tau \in \mathit{Out}} \tau \qquad \text{at } X_{\tau_{in}} \cdot r$$
- Generate $M^{(\infty)}_{\mathit{GPEPA}}$ and its PCTMC $\boldsymbol{X}_{\mathit{GPEPA}}(t), \mathcal{C}_{\mathit{GPEPA}}, \boldsymbol{X}_{\boldsymbol{0},\boldsymbol{T}}$ ◁ **External transitions**
- **foreach** *transition specification* $(T, \mathit{In}, \mathit{Out})$ *in* $\mathbb{E}$ **do**
  - Find a transition class c in $\mathcal{C}_{\mathit{GPEPA}}$ corresponding to T
  - Define $r'_c(\boldsymbol{X})$ from $r_c(\boldsymbol{X}_{\mathit{GPEPA}})$ by replacing elements of $\boldsymbol{X}_{\mathit{GPEPA}}$ indexed by ${}^{\tau,\Gamma}P^{\underline{S}}$ with elements of $\boldsymbol{X}$ indexed by τ.
  - Add the transition class below to $\mathcal{C}$:
    $$\textstyle\sum_{\tau \in \mathit{In}} \tau \to \sum_{\tau \in \mathit{Out}} \tau \qquad \text{at } r'_c(\boldsymbol{X})$$

Populations

The output from Algorithm 1 contains a model $M^{(\infty)}$ which enumerates all the possible transactions in M. Let $\mathbb{T}$ be the set of all such transactions. The population vector is $\boldsymbol{X}(t) \in \mathbb{Z}_+^{|\mathbb{T}|}$. We will index the individual populations in this vector with the corresponding transactions.

Transition classes

Internal transition classes do not depend on the rest of the model. We can directly use the triples from $\mathbb{I}$ to construct the corresponding transition classes in $\mathcal{C}$. The difference vectors are given by the single incoming transaction τ_{in} and a multiset of outgoing transactions Out in each triple in

$([\![\tau_{in}]\!], Out, r) \in \mathbb{I}$. Each instance of τ_{in} can perform the internal communication independently at a rate r and so the resulting transition class is

$$\tau_{in} \to \sum_{\tau \in In} \tau \qquad \text{at } X_{\tau_{in}} \cdot r$$

External transition classes are a result of agents communicating between different transactions. Such communication can be defined by the auxiliary GPEPA model used in Algorithm 1. The same model can be used to define the rates in the external transition classes. Define a model $M^{(\infty)}_{GPEPA}$ as in Section 7.3.1.1 and derive its PCTMC $(\boldsymbol{X}_{GPEPA}(t), \mathcal{C}_{GPEPA}, \boldsymbol{X}_{\boldsymbol{T},\boldsymbol{0}})$ using the GPEPA semantics. Each triple $(U, In, Out) \in \mathbb{E}$ clearly corresponds to a transition class in the GPEPA model, $c \in \mathcal{C}_{GPEPA}$. Because the socket instructions in the GPEPAc model only determine what happens after each transition, the rate function $r_c(\boldsymbol{X}_{GPEPA})$ can be used as the rate of the GPEPAc transition class. Construct $r'_c(\boldsymbol{X})$ by replacing all occurrences of $\boldsymbol{X}_{GPEPA}$ elements indexed with ${}^{\tau,\Gamma}P^{\underline{S}}$ by respective elements X_τ of $\boldsymbol{X}$. The transition class corresponding to the triple thus becomes

$$\sum_{\tau \in In} \tau \to \sum_{\tau \in Out} \tau \qquad \text{at } r'_c(\boldsymbol{X})$$

Initial populations

The initial populations $\boldsymbol{X_0}$ are simply obtained as the number of occurrences of each transaction in the model M. Define a counting function $\#_{G:P^{\underline{S}}(\Gamma)}(\tau)$ as the number of occurrences of the agent $G : P^{\underline{S}}(\Gamma)$ in a transaction τ. Similarly, define $\#_{G:\tau \ni P^{\underline{S}}(\Gamma)}(M)$ as the number of occurrences of the agent $\tau \ni P^{\underline{S}}(\Gamma)$ in a group G in a model M. The count of a transaction τ in a model M is then defined as

$$\#_\tau(M) = \frac{\#_{G:\tau \ni P^{\underline{S}}(\Gamma)}(M)}{\#_{G:P^{\underline{S}}(\Gamma)}(\tau)}$$

which has to be equal for all agents $G : P^{\underline{S}}(\Gamma) \in \tau$. The initial populations are then set as

$$X_{0,\tau} = \#_\tau(M) \qquad \tau \in M^{(\infty)}.$$

> Algorithm 1 eventually obtains the following transactions in the client–server model:
>
> $$\tau^j_k = [\![\mathbf{Clients} : \mathit{Client\_wait}^{\underline{s}}_j(\underline{s} \mapsto \alpha), \mathbf{Servers} : \mathit{Server\_get}^{\underline{c}_k}(\underline{c}_k \mapsto \alpha)]\!]$$
> $$\tau^{ij}_{12} = [\![\mathbf{Clients} : \mathit{Client\_wait}^{\underline{s}}_i(\underline{s} \mapsto \alpha_1), \mathbf{Clients} : \mathit{Client\_wait}^{\underline{s}}_j(\underline{s} \mapsto \alpha_2),$$
> $$\mathbf{Servers} : \mathit{Server\_get}^{\underline{c}_1,\underline{c}_2}(\underline{c}_1 \mapsto \alpha_1, \underline{c}_2 \mapsto \alpha_2)]\!]$$

where $i, j, k \in \{1, 2\}$ and single agent transactions (with abbreviations shown under each transaction)

$$\underbrace{[\![\textbf{Clients} : \mathit{Client}_i]\!]}_{C_i}, \underbrace{[\![\textbf{Clients} : \mathit{Client\_think}_i]\!]}_{C_{ti}},$$
$$\underbrace{[\![\textbf{Servers} : \mathit{Server}]\!]}_{S}, \underbrace{[\![\textbf{Servers} : \mathit{Server\_broken}]\!]}_{S_b}$$

The complete model $M^{(\infty)}$ contains all the agents in these transactions. The resulting PCTMC consists of 14 populations, one for each transaction. The possible internal transition classes are

$$\begin{aligned}
\tau_k^i &\to S + C_{ti} && \text{at } \tau_k^i \cdot r_{data,i} \\
\tau_{12}^{ij} &\to \tau_1^i + C_{tj} && \text{at } \tau_{12}^{ij} \cdot r_{data,j} \\
\tau_{12}^{ij} &\to \tau_2^j + C_{ti} && \text{at } \tau_{12}^{ij} \cdot r_{data,i} \\
\tau_{12}^{ij} &\to S_b + C_i + C_j && \text{at } \tau_{12}^{ij} \cdot r_{break}
\end{aligned}$$

The external transition classes are

$$\begin{aligned}
S + C_i &\to \tau_1^i && \text{at } \frac{S}{r_S(\boldsymbol{X})} \cdot \frac{C_i}{C_1 + C_2} \cdot \min(r_S(\boldsymbol{X}), C_1 + C_2) \cdot r_{request} \\
\tau_1^i + C_j &\to \tau_{12}^{ij} && \text{at } \frac{\tau_1^i}{r_S(\boldsymbol{X})} \cdot \frac{C_j}{C_1 + C_2} \cdot \min(r_S(\boldsymbol{X}), C_1 + C_2) \cdot r_{request} \\
\tau_2^i + C_j &\to \tau_{12}^{ji} && \text{at } \frac{\tau_2^i}{r_S(\boldsymbol{X})} \cdot \frac{C_j}{C_1 + C_2} \cdot \min(r_S(\boldsymbol{X}), C_1 + C_2) \cdot r_{request} \\
C_{ti} &\to \mathit{Client}_i && \text{at } C_{ti} \cdot r_{think} \\
S &\to S_b && \text{at } S \cdot r_{break} \\
S_b &\to S && \text{at } S_b \cdot r_{reset}
\end{aligned}$$

where $i, j \in \{1, 2\}$ and

$$r_S(\boldsymbol{X}) = S + \sum_{i,k \in \{1,2\}} \tau_k^i$$

The initial populations are zero except for $X_{0,C_i} = n_i$, $X_{0,S} = n_S$.

7.4 Case study: A large scale computing cluster

In this section, we demonstrate GPEPAc on a model of a computing cluster with nodes capable of concurrently executing a number of jobs. Both job and node instances come from different classes. A scheduling policy determines the allocation of jobs to nodes.

We show how to apply the GPEPAc semantics to produce a PCTMC representing the model. We augment the PCTMC with a continuous variable capturing the total cluster energy consumption

to obtain a *h*PCTMC model (Chapter 6). We present a number of numerical examples obtained by solving the *h*PCTMC model with the efficient ODE analysis from Section 6.3. We look at the performance–energy trade-off from Section 5.5 and show how to find a combination of scheduling policy and cluster configuration that simultaneously satisfies an SLA for each job class and minimises the total energy consumption.

7.4.1 The model

A hypothetical computing cluster consists of a large number of nodes of different types, capable of processing a large number of job requests from users with different priorities. The requests and node allocations are handled by scheduler threads with various scheduling policies. Figure 7.2 shows an overview of the system.

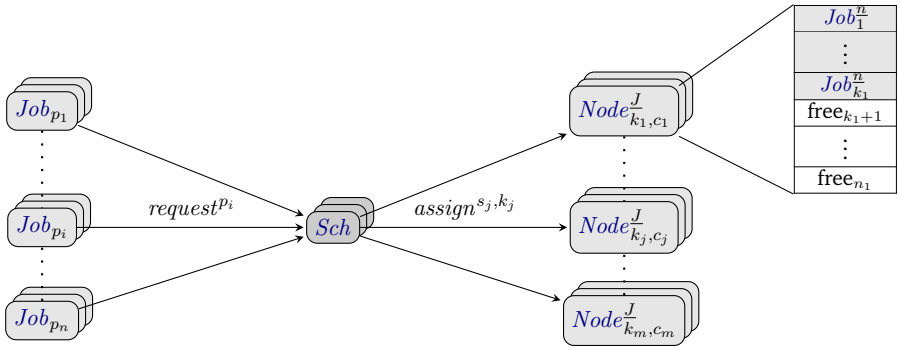


Figure 7.2: Overview of the cluster model.

The subscripts p_i for jobs stand for different job priorities. The subscripts c_j for nodes stand for different node classes and the subscripts k_j for the current number of jobs already assigned to the nodes, as shown by the detail of the class c_1 with k_1 assigned jobs out of the total capacity n_1.

Nodes

Nodes come from different classes, drawn from a set $\mathcal{C}_{Node}$, for example differing in their processing speed or failure rates. Each node from a class $c \in \mathcal{C}_{Node}$ is capable of simultaneously serving n jobs, up to $capacity(c) \in \mathbb{N}$, through a set of sockets $\underline{J_n}$, one for each assigned job. Each node can respond to processing requests from the allocated jobs on a channel linking to the job, with rate $r^{c,n}_{proc}$ depending on the current node occupancy n. Each node advertises an empty processing slot and its current class c and occupation n (when $n < capacity(c)$) via the action $assign^{c,n}$. A new job can be assigned to the node via synchronisation on this action. The node selects an available socket $\underline{j}_i$ with the smallest index i. Jobs can be de-allocated from nodes via the end action, with nodes closing the respective socket. Finally, a node can break, announcing the failure on all its channels and switching to a state where it has to be repaired. The definition of nodes in GPEPAc is:

$$
\begin{aligned}
Node_{0,c} &\stackrel{def}{=} (assign^{c,0} \text{ get } \underline{j}_1, r^{c,0}_{assign}).Node^{\underline{j}_1}_{1,c} \\
Node^{J_n}_{n,c} &\stackrel{def}{=} \sum_{\underline{j} \in \underline{J_n}} (proc \text{ receive on } \underline{j}, r^{c,n}_{proc}).Node^{J_n}_{n,c} + \sum_{\underline{j} \in \underline{J_n}} (end \text{ receive on } \underline{j}, \top).Node^{J_n \setminus \{\underline{j}\}}_{n-1,c} \\
&\quad + (assign^{c,n} \text{ get } \underline{j}_{\min}, r^{c,n}_{assign}).Node^{J_n \cup \{\underline{j}_{\min}\}}_{n+1,c}
\end{aligned}
$$

$$
\begin{aligned}
&+ (break \,\mathsf{send\ on}\, \underline{J_n}, r^c_{break}).Node_{broken,c} && 1 \le n < capacity(c)\\
Node^{\underline{J_m}}_{m,c} &\stackrel{def}{=} \sum_{\underline{j} \in \underline{J_m}} (proc \,\mathsf{receive\ on}\, \underline{j}, r^{c,m}_{proc}).Node^{\underline{J_m}}_{m,c} + \sum_{\underline{j} \in \underline{J_m}} (end \,\mathsf{receive\ on}\, \underline{j}, \top).Node^{\underline{J_m}\setminus\underline{j}}_{m-1,c}\\
&+ (break \,\mathsf{send\ on}\, \underline{J_m}, r^c_{break}).Node_{broken,c} && m = capacity(c)\\
Node_{broken,c} &\stackrel{def}{=} (reset, r^c_{reset}).Node_{0,c}
\end{aligned}
$$

where $c \in \mathcal{C}_{Node}$ and $\underline{J_n} = \{\underline{j}_{i_1}, \ldots, \underline{j}_{i_n}\} \subseteq \{\underline{j}_1, \ldots, \underline{j}_m\}$, $\underline{j}_{\min} = \underline{j}_{\min(\{1,\ldots,m\}\setminus\{i_1,\ldots,i_n\})}$.

Jobs

The cluster processes jobs with multiple priorities from the set $\mathcal{C}_{Job}$. Each job requests a node from the scheduler described below. The scheduler creates a new channel and sends a copy to the requesting job. After selecting a suitable node, the scheduler forwards the channel, creating a link between the job and the node. Each job from a priority class p runs through a sequence of processing commands with length $states(p) \in \mathbb{N}$, followed by a termination command end:

$$
\begin{aligned}
Job_p &\stackrel{def}{=} (request^p \,\mathsf{get}\, \underline{n}, r^p_{request}).Job^{\underline{n}}_{1,p}\\
Job^{\underline{n}}_{i,p} &\stackrel{def}{=} (proc \,\mathsf{send\ on}\, \underline{n}, \top).Job^{\underline{n}}_{i+1,p} + (break \,\mathsf{receive\ on}\, \underline{n}, \top).Job_p\\
Job^{\underline{n}}_{states(p),p} &\stackrel{def}{=} (end \,\mathsf{send\ on}\, \underline{n}, r^p_{end}).Job_p
\end{aligned}
$$

where $p \in \mathcal{C}_{Job}$, $1 \le i < states(p)$.

Schedulers

Schedulers provide an interface between unassigned jobs and nodes with empty slots. They respond to client requests, aware of their priority through the $request^p$ action, switching into the state Sch_p. In each such state, schedulers try to assign the job to an available node according to a *scheduling policy*: the node class c and current job occupancy n are determined via probabilistic weighting of the rate parameters $r^{c,n}_{assign,p}$ for synchronisation on the $assign^{c,n}$ actions. The formal definition is:

$$
\begin{aligned}
Sch &\stackrel{def}{=} \sum_p (request^p \,\mathsf{init}\, \underline{j}, r_{request}).Sch^{\underline{j}}_p\\
Sch^{\underline{j}}_p &\stackrel{def}{=} \sum_{c \in \mathcal{C}_{Node}} \sum_{n=0}^{capacity(c)-1} (assign^{c,n} \,\mathsf{forward}\, \underline{j}, r^{c,n}_{assign,p}).Sch
\end{aligned}
$$

where $p \in \mathcal{C}_{Job}$.

Model

The final model consists of n_p copies of p priority jobs, n_c copies of nodes from each class c and n_S copies of the scheduler:

$$
\begin{gathered}
\mathbf{Jobs}\{Job_{p_1}[n_{p_1}] \parallel \cdots \parallel Job_{p_n}[n_{p_n}]\}\\
\bowtie_{L_1}\\
\Big(\mathbf{Schedulers}\{Sch[n_S]\} \bowtie_{L_2} \mathbf{Nodes}\{Node_{0,c_1}[n_{c_1}] \parallel \cdots \parallel Node_{0,c_m}[n_{c_m}]\}\Big)
\end{gathered}
$$

where $\mathcal{C}_{Node} = \{c_1, \ldots, c_m\}$, $\mathcal{C}_{Job} = \{p_1, \ldots, p_n\}$ and the cooperation sets are

$$L_1 = \{request^{p_1}, \ldots, request^{p_n}\}$$
$$L_2 = \{assign^{c,n} \mid c \in \mathcal{C}_{Node}, 0 \leq n < capacity(c)\}$$

7.4.2 Resulting PCTMC

We follow the definition of PCTMC semantics of GPEPAc from Section 7.3 and describe the underlying PCTMC of the cluster model. We stress that we show the full PCTMC for illustration purposes only – in practical cases it is automatically generated from the GPEPAc description and the complexity is hidden from the modeller.

Possible transactions

There are four types of agents without a socket – idle schedulers, unassigned jobs, nodes and broken nodes:

$$\underbrace{[\![Sch]\!]}_{S}, \quad \underbrace{[\![Job_p]\!]}_{J_p} \qquad p \in \mathcal{C}_{Job}$$
$$\underbrace{[\![Node_{0,c}]\!]}_{N_{0,c}} \quad \underbrace{[\![Node_{broken,c}]\!]}_{N_{b,c}} \qquad c \in \mathcal{C}_{Node}$$

For brevity, we will use the abbreviations shown under each transaction.

Channels are initialised by the scheduler agents. Each scheduler gets linked to a single job, resulting in transactions of the form

$$\underbrace{[\![Job^n_{1,p}(\{\underline{n} \mapsto \alpha\})Sch^j_p(\{\underline{j} \mapsto \alpha\})]\!]}_{[\![J_p S]\!]} \qquad p \in \mathcal{C}_{Job}$$

There are $|\mathcal{C}_{Job}|$ such transactions.

A scheduled job can be assigned to a node. Each node can already be linked, by different channels, to a number of jobs given by the nodes capacity. The jobs can be in any of their possible states. Therefore all the remaining transactions are of the form

$$\underbrace{[\![Job^n_{k_1,p_1}(\{\underline{n} \mapsto \alpha_1\}) \ldots Job^n_{k_n,p_n}(\{\underline{n} \mapsto \alpha_n\})Node^{J_n}_{n,c}(\{\underline{j}_{i_k} \mapsto \alpha_k\}^n_{k=1})]\!]}_{[\![i_1:J_{k_1,p_1},\ldots,i_k:J_{k_n,p_n}N_{n,c}]\!]} \qquad c \in \mathcal{C}_{Node}$$

where $n \leq capacity(c)$, $p_i \in \mathcal{C}_{Job}$ and $1 \leq k_i \leq states(p_i)$ for $i = 1, \ldots, n$. We can note that the socket indexes $i_1, \ldots, i_k$ in the transaction form above do not affect the behaviour of the transactions. At this stage, we perform an obvious aggregation and will only remember the total number of occupied node sockets. We will write

$$[\![J_{k_1,p_1}, \ldots, J_{k_n,p_n}N_{n,c}]\!]$$

for the transactions under such aggregation.

In total, there are

$$N_1 = \sum_{c \in C_{Node}} \sum_{n=1}^{capacity(c)} \left(\sum_{c \in C_{Job}} states(p) \right)^n$$

such transactions. Therefore, the state space of the model is $\boldsymbol{X}(t) \in \mathbb{Z}_+^N$ where $N = 1 + 2 \cdot |C_{Job}| + 2 \cdot |C_{Node}| + N_1$.

Transition classes

There are three types of external transition classes in this model. Job request transitions are the same as they would be in standard GPEPA, with a difference that the right-hand side consists of a new transaction between a job and a scheduler instead of two independent agents. The first type of transition classes is of the form

$$J_p + S \to [\![J_{1,p} S_p]\!] \qquad (request^p)$$

for all $p \in \mathcal{C}_{Job}$.

In transitions corresponding to the $assign^{s,n}$ actions, a transaction of an unassigned job with a scheduler $[\![J_{1,p} S_p]\!]$ is synchronised with a transaction consisting of a node $N_{n,c}$ linked to n jobs $J_{j_{k_i},p_i}$, $1 \le i \le n$. The job is assigned to the node by forwarding the channel after which the scheduler is freed. This results in transition classes of the form

$$[\![J_{1,p} S_p]\!] + [\![J_{k_1,p_1} \cdots J_{k_n,p_n} N_{n,c}]\!] \to [\![J_{1,p} J_{k_1,p_1} \cdots J_{k_n,p_n} N_{n+1,c}]\!] + S \qquad (assign^{c,n}_{p,\boldsymbol{p},\boldsymbol{k}})$$

for any $p \in \mathcal{C}_{Job}$, $c \in \mathcal{C}_{Node}$, $n < capacity(c)$ and $p_1, \ldots, p_n \in \mathcal{C}_{Job}$, $k_i < states(p_i)$ for all $i = 1, \ldots, n$.

Finally, a broken node can be reset

$$N_{broken,c} \to N_{0,c} \qquad (reset^c)$$

There are three types of internal transition classes. Processing events occur within all the possible transactions of jobs assigned to nodes:

$$[\![J_{k,p} \cdots N_{n,c}]\!] \to [\![J_{k+1,p} \cdots N_{n,c}]\!] \qquad (proc^{c,n}_{k,p,\boldsymbol{k},\boldsymbol{p}})$$

for $p \in \mathcal{C}_{Job}$ and $j < states(p)$, $c \in \mathcal{C}_{Node}$ and the pair of "$\cdots$" standing for the other $n-1$ jobs $J_{k_1,p_1}, \ldots, J_{k_{n-1},p_{n-1}}$ assigned to the node. Similarly for the end transitions:

$$[\![J_{states(p),p} \cdots N_{n,c}]\!] \to [\![\cdots N_{n-1,c}]\!] + J_p \qquad (end^{n,c}_{p,\boldsymbol{k},\boldsymbol{p}})$$

In case of node failures, the node $N_{n,c}$ notifies all the n allocated jobs and the transaction breaks into n jobs and a broken node:

$$[\![J_{k_1,p_1} \cdots J_{k_n,p_n} N_{n,c}]\!] \to J_{p_1} + \cdots + J_{p_n} + N_{broken,c} \qquad (break^{n,c}_{\boldsymbol{p},\boldsymbol{k}})$$

for $p_i \in \mathcal{C}_{Job}$, $k_i \le states(p_i)$ for $i = 1, \ldots, n$ and $c \in \mathcal{C}_{Node}$, $n \le capacity(c)$.

Rates

Transition class rate functions are based on the original GPEPA semantics, derived via the auxiliary model $M^{(\infty)}_{GPEPA}$ in Algorithm 2. The rate functions for transition classes of the type $(request^p)$ are given directly by GPEPA semantics as the event is a result of cooperation between two standard GPEPA agents:

$$r_{request^p}(\boldsymbol{X}(t)) = \frac{r^p_{request} J_p(t)}{R_{request,Job}(\boldsymbol{X}(t))} \cdot \min\left(R_{request,Job}(\boldsymbol{X}(t)), r_{request} \cdot S(t)\right) \tag{7.3}$$

where

$$R_{request,Job}(\boldsymbol{X}(t)) = \sum_{q \in \mathcal{C}_{Job}} r^q_{request} \cdot J_q(t)$$

Transitions classes of type $(assign^{c,n}_{p,\boldsymbol{p},\boldsymbol{k}})$ involve a cooperation of a job–scheduler transaction and a node–jobs transaction. If we apply GPEPA semantics treating the transactions as if they were the PEPA agents executing the action, we get the rate:

$$\begin{aligned} r_{assign^{c,n}_{p,\boldsymbol{p},\boldsymbol{k}}}(\boldsymbol{X}(t)) = {} & \frac{r^{c,n}_{assign,p} \cdot [\![J_{1,p} S_p]\!](t)}{R^{c,n}_{assign,Sch}(\boldsymbol{X}(t))} \cdot \frac{r^{c,n}_{assign} \cdot [\![J_{j_1,p_1} \cdots J_{j_n,p_n} N_{n,c}]\!](t)}{R^{c,n}_{assign,Node}(\boldsymbol{X}(t))} \\ & \cdot \min(R^{c,n}_{assign,Sch}(\boldsymbol{X}(t)), R^{c,n}_{assign,Node}(\boldsymbol{X}(t))) \end{aligned} \tag{7.4}$$

where

$$\begin{aligned} R^{c,n}_{assign,Sch}(\boldsymbol{X}(t)) &= \sum_{q \in \mathcal{C}_{Job}} r^{c,n}_{assign,q} [\![J_{1,q} S_q]\!](t) \\ R^{c,n}_{assign,Node}(\boldsymbol{X}(t)) &= \sum_{q_i \in \mathcal{C}_{Job}, k_i < states(q_i)} r^{c,n}_{assign} [\![J_{k_1,q_1} \cdots J_{k_n,q_n} N_{n,c}]\!](t) \end{aligned}$$

These are the only two types of external transition classes resulting from agent cooperation. All the other transitions are internal to transactions. Treating these in the same way as internal transitions of standard PEPA agents, we get rates of the form $r \times [\![\cdots]\!](t)$. In case of transition classes of the type $(proc^{c,n}_{k,p,\boldsymbol{k},\boldsymbol{p}})$, the rates are

$$r_{proc^{c,n}_{p,\boldsymbol{p},\boldsymbol{k}}}(\boldsymbol{X}(t)) = r^{c,n}_{proc} \cdot [\![J_{j,p} \cdots N_{n,c}]\!](t) \tag{7.5}$$

Similarly for $(end^{n,c}_{p,\boldsymbol{k},\boldsymbol{p}})$ transition classes:

$$r_{end^{n,c}_{p,\boldsymbol{p},\boldsymbol{k}}}(\boldsymbol{X}(t)) = r_{end,p} \cdot [\![J_{s(p),p} \cdots N_{n,s}]\!](t) \tag{7.6}$$

For $(break^{n,c}_{\boldsymbol{p},\boldsymbol{k}})$ transitions the rates are:

$$r_{break^{n,c}_{\boldsymbol{p},\boldsymbol{k}}}(\boldsymbol{X}(t)) = r^{s,n}_{break} \cdot [\![J_{j_1,p_1} \cdots J_{j_n,p_n} N_{n,s}]\!](t) \tag{7.7}$$

Table 7.1: Rates used in the cluster model.

	$p = L$			$p = H$		
n_p	200			50		
	$c = slow$			$c = fast$		
r^c_{repair}	0.5			0.5		
n_c	40			32		
	$n = 0$	$n = 1$	$n = 2$	$n = 0$	$n = 1$	$n = 2$
$r^{c,n}_{assign}$	1.0	1.0	–	2.0	2.0	–
$r^{c,n}_{proc}$	–	0.8	0.5	–	4.0	3.0
$r^{c,n}_{fail}$	0.1	0.4	0.6	0.1	0.2	0.3
$r^{c,n}_{energy}$	0.05	0.1	0.15	0.05	0.3	0.5

and for $(reset^c)$:

$$r_{reset^c}(\boldsymbol{X}(t)) = r^c_{reset} \cdot N_{broken,c}(t) \tag{7.8}$$

Energy consumption

Similar to the client–server model, Section 5.2, we can argue that the total energy consumption by all nodes in the cluster can be expressed as a linear combination of accumulated transaction populations. We can extend the model PCTMC with a continuous variable capturing this reward, and obtain a *h*PCTMC from Section 6:

$$\mathcal{A}_{\text{energy}}(t) = \sum_{c \in \mathcal{C}_{Node}} \sum_{n=0}^{capacity(c)} r^{c,n}_{\text{energy}} \sum_{\substack{p_1,\ldots,p_n \in \mathcal{C}_{Job} \\ 0<j_i \le states(p_i),\ i=1,\ldots,n}} \int_0^t [\![J_{j_1,p_1} \cdots J_{j_n,p_n} N_{n,c}]\!](u)\mathrm{d}u \tag{7.9}$$

for node energy consumption rates $r^{c,n}_{\text{energy}}$ for each node of class c, serving n jobs.

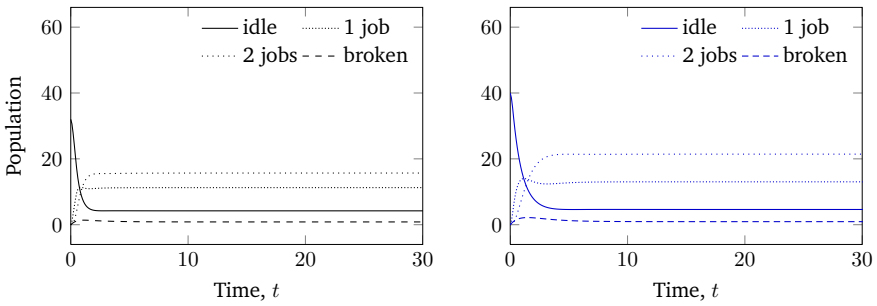
7.4.3 Numerical examples

We examine a specific instance of the cluster model with two job priorities, *low* and *high*, $\mathcal{C}_{Job} = \{L, H\}$, having two and three processing stages respectively, that is $states(L) = 2$, $states(H) = 3$. There are two node classes, *slow* and *fast*, $\mathcal{C}_{Node} = \{slow, fast\}$, each capable of processing two jobs at a time, that is $capacity(c) = 2$ for $c \in \mathcal{C}_{Node}$. We assume that the fast nodes are around five times faster than the slow nodes – the rate of processing a single job is 4.0 and 0.8 respectively. Additionally, we assume that the fast nodes cope better when concurrently serving two jobs – the rate of processing one of two concurrently running jobs is 3.0 and 0.5 respectively. The increased performance is at the expense of energy consumption. We assume that the fast servers consume around 3 times as much energy per unit of time as slow servers. Table 7.1 shows all constants in the numerical example, except for the scheduling weights which are defined as part of scheduling policies.

We examine a range of different scheduling policies. Table 7.2b shows the parameters for a uniform policy which assigns jobs to servers regardless of the job priority and node class and occupancy. We look at the effects of scheduling policies on response times for jobs and on the total

energy consumption of all nodes. We use the ODE analysis to compute passage time probabilities of a job from each class finishing for the first time (executing the first end action) [4]. This technique is directly applicable to the GPEPAc model, as we only replace the sequential job agents. We use the continuous variable from Equation 7.9 to compute the total energy consumption of nodes.

We apply the ODE analysis to the underlying PCTMC to obtain means of agent populations, passage time probabilities for each job priority and the total energy consumption. The PCTMC has 273 populations and 736 transition classes. Additionally we capture the energy consumption as a continuous variable as described in Section 6.2. The ODE analysis generates a system of 274 ODEs. It takes around 0.1 seconds on to numerically solve this system until time t_f that is sufficiently large to approximate a steady state in the model ($t_f = 30$ in the examples below) a standard desktop computer . For example, Figure 7.3 shows the mean populations of nodes in different states – idle, broken and in a transaction with one or two jobs.



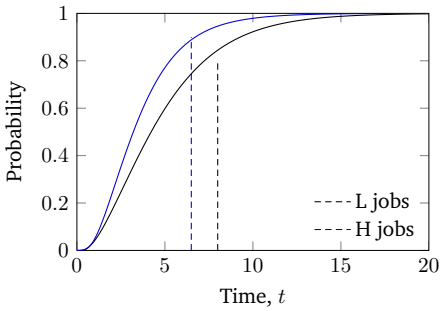
(a) Fast nodes (b) Slow nodes

Figure 7.3: Population of different node classes and occupancies over time.

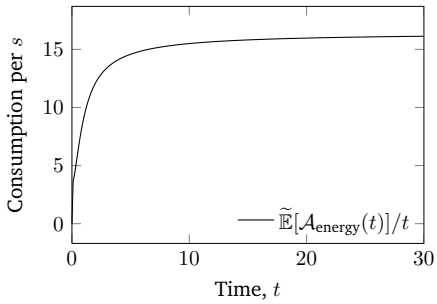
We examine the trade-off between performance and energy consumption, in the same fashion as in Section 5.5. We define a service level agreement for each job class. For example, we require that a low priority job finishes within 8.0 seconds with probability at least 0.8 and a high priority job finishes within 6.5 seconds with probability at least 0.9. We aim to minimise the steady state rate of energy consumption, that is the limit of $\mathbb{E}[\mathcal{A}_{\text{energy}}(t)]/t$ as $t \to \infty$, while satisfying the given SLAs. Figure 7.4 plots the passage time probability and the energy consumption per time for two different system configurations. When there are 36 slow nodes and 34 fast nodes, SLAs for both low and high priority jobs are satisfied and the energy consumption rate is around 16 units per unit of time. When the number of nodes decreases to 26 and 24 respectively, the consumption decreases to around 11 units per unit of time, but both SLAs are violated.

7.4.4 Optimising the cluster configuration

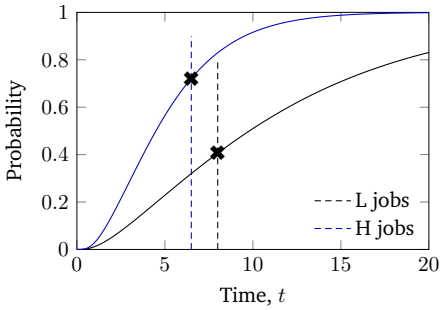
Because of the low computational cost of the ODE analysis of PCTMCs, we can afford to explore large number of model parameter combinations. In case of the cluster model, we can assume that all the rates in Table 7.1, except for the number of nodes n_{slow} and n_{fast}, cannot be modified by the system providers as they represent performance and energy consumption parameters of the available hardware and the expected job workload. We assume that system providers can choose the number of nodes in the system and the parameters of the scheduling policy.



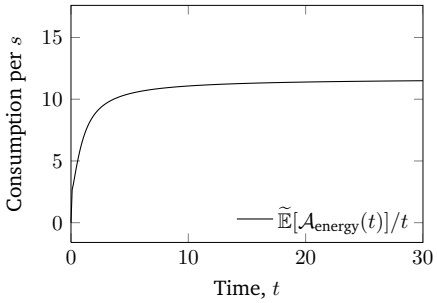
(a) $n_{slow} = 36$, $n_{fast} = 34$: SLA compliance



(b) $n_{slow} = 36$, $n_{fast} = 34$: Energy consumption



(c) $n_{slow} = 26$, $n_{fast} = 24$: SLA compliance



(d) $n_{slow} = 26$, $n_{fast} = 24$: Energy consumption

Figure 7.4: Response time probabilities for a low and high priority job and the rate of energy consumption of the cluster. The dashed lines visualise the SLA thresholds – 6.5 seconds with probability at least 0.9 for high priority jobs and 8 seconds with probability at least 0.8 for low priority jobs.

We first take three scheduling policies and find the optimal number of nodes such that the energy consumption is minimised and both SLAs are satisfied. For each policy, we evaluate all cluster configurations with up to 70 nodes from each of the two classes. For each configuration, we numerically solve the ODE system generated from the *h*PCTMC corresponding to the model and calculate the passage time probabilities and the steady state energy consumption.

The first policy dedicates one server class to each job priority – slow nodes to low priority jobs and fast nodes to high priority jobs, with rates in Table 7.2a. In such case the SLAs are never simultaneously satisfied in any configuration. As the second policy, we take the uniform policy where jobs are assigned to nodes regardless of priorities and node classes and occupancies. The rates are listed in Table 7.2b. Figure 7.5a visualises the SLA satisfaction and energy consumption across all configurations. In this particular case, the lowest energy consumption of 16.59 units is achieved when $n_{slow} = 10$ and $n_{fast} = 44$.

Because the SLA for high priority jobs is more strict, i.e. the time by which jobs are required to finish is lower and the probability higher, it makes sense for the scheduling policy to give priority to these jobs. This can be achieved by setting some of the rates $r^{c,n}_{assign,H}$ higher than the respective rates $r^{c,n}_{assign,L}$. For example, the scheduler can make it twice as likely to assign a higher priority job onto a fast node than a lower priority one and equally likely for slow nodes:

$$r^{fast,0}_{assign,H} = 2 \cdot r^{fast,0}_{assign,L} = 4.0 \qquad r^{slow,0}_{assign,H} = r^{slow,0}_{assign,L} = 2.0$$

Table 7.2: Values of $r^{c,k}_{assign,p}$ for different scheduling policies in the case study.

(a) Dedicated nodes for each class

	$c = slow$		$c = fast$	
p	$k = 0$	$k = 1$	$k = 0$	$k = 1$
L	2.0	1.0	0	0
H	0	0	4.0	2.0

(b) Uniform policy

	$c = slow$		$c = fast$	
p	$k = 0$	$k = 1$	$k = 0$	$k = 1$
L	2.0	2.0	2.0	2.0
H	2.0	2.0	2.0	2.0

(c) Priority weighted policy

	$c = slow$		$c = fast$	
p	$k = 0$	$k = 1$	$k = 0$	$k = 1$
L	2.0	1.0	1.0	0.5
H	2.0	1.0	4.0	2.0

(d) Optimal policy for $n_{slow} = 24$, $n_{fast} = 58$

	$c = slow$		$c = fast$	
p	$k = 0$	$k = 1$	$k = 0$	$k = 1$
L	1.2	0.6	0.8	0.4
H	0.8	0.4	3.6	1.8

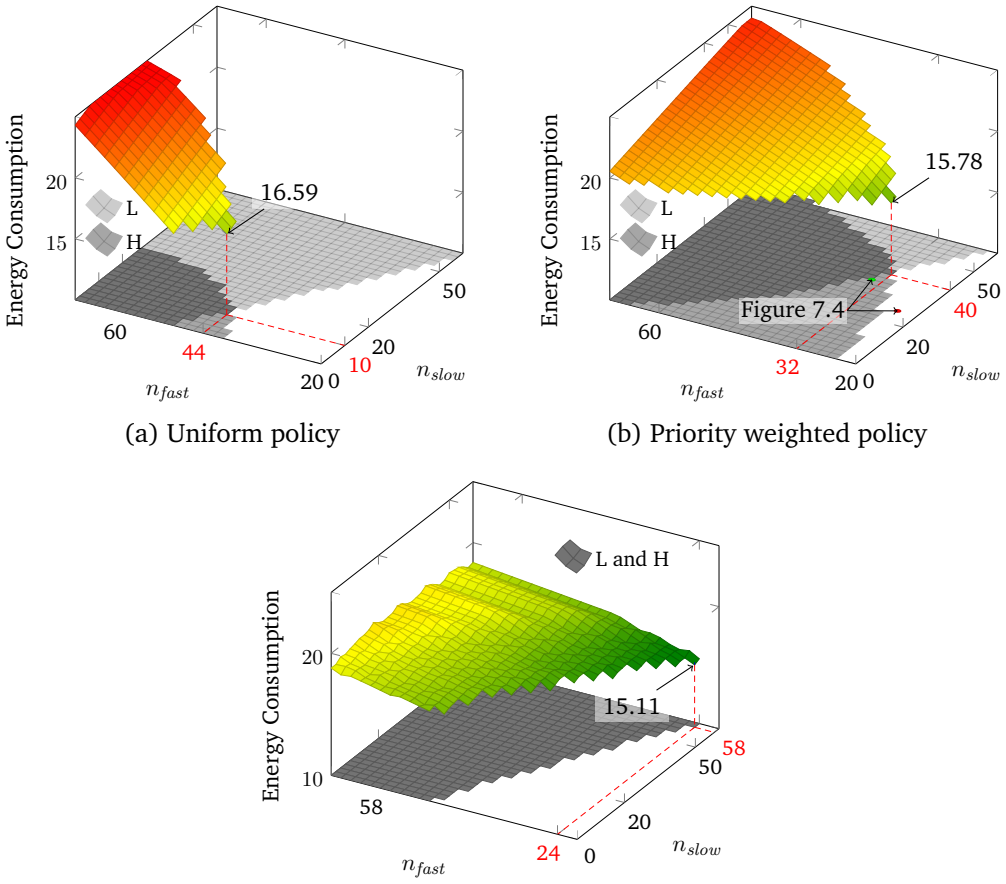
To make sure idle nodes are likely to be assigned before second jobs are assigned to already occupied nodes, we can set the rate of assignment higher (say twice) for nodes with no jobs than for ones with one job

$$r^{c,0}_{assign,p} = 2 \times r^{c,1}_{assign,p}$$

Figure 7.5b shows the parameter exploration for this scheduling policy. The minimum energy consumption 15.78 units per unit time is obtained when there are 40 slow nodes and 32 fast nodes. This minimum is lower than in case of the uniform scheduling policy.

So far, we have fixed a scheduling policy and tried to find the node configuration minimising energy consumption. Due to the low computational cost of ODE analysis, we can also afford to vary the possible scheduling parameters for each cluster configuration. In practice, this would represent different software settings which are arguably easier to set than hardware configurations. We generalise the third scheduling policy. We fix the total assignment rate for idle nodes for low and high priority jobs, to say 3.0 and 6.0 respectively, and set the rate to occupied nodes to be a half of the rate for idle nodes. For each job priority we vary the proportion of the total rate being assigned to fast nodes. For example in the scheduling policy above, we have the rate 2.0 out of 3.0 for low priority jobs to slow servers and the remaining rate 1.0 to fast servers and 4.0 out of 6.0 for high priority jobs to fast nodes and the remaining 2.0 to slow nodes .

Figure 7.5c plots the energy consumption for the best scheduling policy that satisfies both SLAs, if such a policy exists, for each node configuration. The minimum energy consumption of 15.1 units per unit time is obtained when there are 58 fast nodes and 24 slow nodes and is lower than in the fixed scheduling policies above. The particular rates achieving the minimum can be found in Table 7.2d.



(a) Uniform policy

(b) Priority weighted policy

(c) Optimised policy for each configuration

Figure 7.5: Energy consumption and SLA satisfaction for varying cluster configurations under different scheduling policies. Figures (a) and (b) maintain a fixed scheduling policy. Figure (c) shows for each configuration the results of the optimal policy among a range of priority policies.

7.5 Conclusion

In this chapter we have proposed a framework that allows us to concisely capture multi-phase interactions in large-scale population models. We presented GPEPAc, a channel-based extension of the GPEPA process algebra. This extension adds a minimal syntactical layer in which agents can keep track of transaction partners via named sockets. The PCTMC semantics of GPEPAc uses the GPEPA semantics on a sequence of related models to obtain all the possible transaction configurations. A further derived GPEPA model is used to obtain the rates in transition classes of the resulting PCTMC. This approach allows us to apply the efficient ODE analysis on transaction based models and efficiently derive passage times and reward measures in the models. We demonstrated the GPEPAc extension on a model of a heterogeneous computing cluster, with multiple node classes and job priorities. We explored a large number of parameters in order to find the optimal cluster configuration and scheduling policy.

The developments in this chapter are crucial in providing a wider applicability of the scalable analysis of PCTMC models. Apart from the increased expressive power of the GPEPAc process algebra, we are now able to include models where synchronisations have non-exponential

durations. The multi-phase transaction cooperation can be directly used to represent phase-type distributions – we will show an example of a model constructed from real data in Section 9.2.

Chapter 8

GPA – a tool for rapid analysis of PCTMCs

Key contributions		
Implementation of the PCTMC framework		[8, 12]
Multiple supported formalisms	8.2	[2, 5]
Rewards and *h*PCTMC implementation	8.2.4	[11]
Optimisation experiments	8.4.1	[11]

8.1 Introduction

The framework and techniques presented in this thesis provide an efficient way to analyse models where explicit state space approaches require an infeasible amount of computation. However, especially the higher-order ODE analysis requires a derivation of a system of ODEs with a complicated structure. This is even more obvious for PCTMC models described in a concise high-level formalism such as the GPEPA process algebra and the GPEPAc extension from Chapter 7. Obtaining moment approximations from model descriptions manually would be highly impractical and error-prone. Therefore the applicability of the PCTMC framework would be limited without an efficient and extensible implementation.

In this chapter, we present *GPA* – a tool implementing most of the contributions in this thesis. GPA provides a convenient interface for applications of the PCTMC and related techniques to a range of models and also offers a flexible implementation that allows development of further extensions. Initially, GPA was used to investigate the accuracy of higher-order moment approximation for GPEPA by Hayden and Bradley [99], summarised in Section 4.2. Since then, the tool development was interleaved with the theoretical contributions of this thesis. The second iteration of GPA implemented accumulated rewards from Chapter 5, limited to GPEPA models. A later stage consisted of an extension supporting a subset of PCTMCs alongside of GPEPA. When experimenting with different moment closures, Section 4.4 and Section 4.5, we rewrote the core of GPA. All the analyses now work with an intermediate PCTMC representation and different specification languages such as GPEPA and the chemical equation language are realised on a separate layer that generates the PCTMC semantics. We implemented the *h*PCTMC extension from Chapter 6 and features such as parameter exploration for evaluating performance–energy trade-offs. Additionally, the extensible architecture allowed a fast implementation of the spatial process algebra MASSPA [92] and Unified Stochastic Probes [5].

The main distinctive features of GPA are:

General PCTMC framework At the core of GPA is an extensible implementation of the PCTMC framework. Each model is represented in an abstract form as a PCTMC, to which a number of analyses can be applied. GPA implements the mean-field and higher-order moment ODE analyses from Section 3.4. Different moment closure modules are supported, such as an implementation of the min-normal closure from Section 4.4. GPA provides a number of numerical integration algorithms and allows to extract various measures from the numerical solution, such as moments of populations, rewards or passage time probabilities. Additionally, GPA can compute estimates of these metrics from simulation.

High-level specification languages Models in GPA can be specified in the GPEPA process algebra or using a chemical equation style language from Section 3.1. These are part of a layer where additional languages can be added together with an implementation of their PCTMC semantics.

***h*PCTMC extension** Models in any of the supported specification language in GPA can be augmented with continuous variables from the *h*PCTMC extension from Chapter 6. GPA also supports time-dependent rates, as described in Section 6.6, that can be included from external data.

Parameter exploration An analysis of a model in GPA can form a part of a parameter exploration and minimisation experiment. GPA implements minimisation by explicit parameter sweeping as well as by a general approximate global optimisation solver.

Efficient implementation The core of GPA is implemented in the Java programming language. GPA keeps an abstract representation of models and analyses and dynamically generates optimised Java code for numerical computation. Additional modules allow translation to C++ and Matlab.

Figure 8.1 gives an overview of the architecture of GPA. GPA is primarily a command line application, taking as an input a complete model description together with the related analyses. GPA outputs the analysis results to specified files and can additionally visualise the data. Most of the plots in this thesis were produced with data from GPA. The core functionality of GPA can be also easily accessed via a library, making it possible to use its features in larger projects.

In the remainder of this chapter, we describe individual features of GPA. We show examples corresponding to models in this thesis. For each we give relevant portions of the specification code and also show the graphical output when applicable. Full syntax of GPA files can be found in Appendix D.1

8.2 Model syntax

Each GPA input file consists of a single model definition and a number of analyses and secondary experiments. The first section of the file consists of definitions of all the numerical parameters used as rates and initial conditions in the model. For example the following code defines parameters in the client–server model from Table 4.1, Model A:

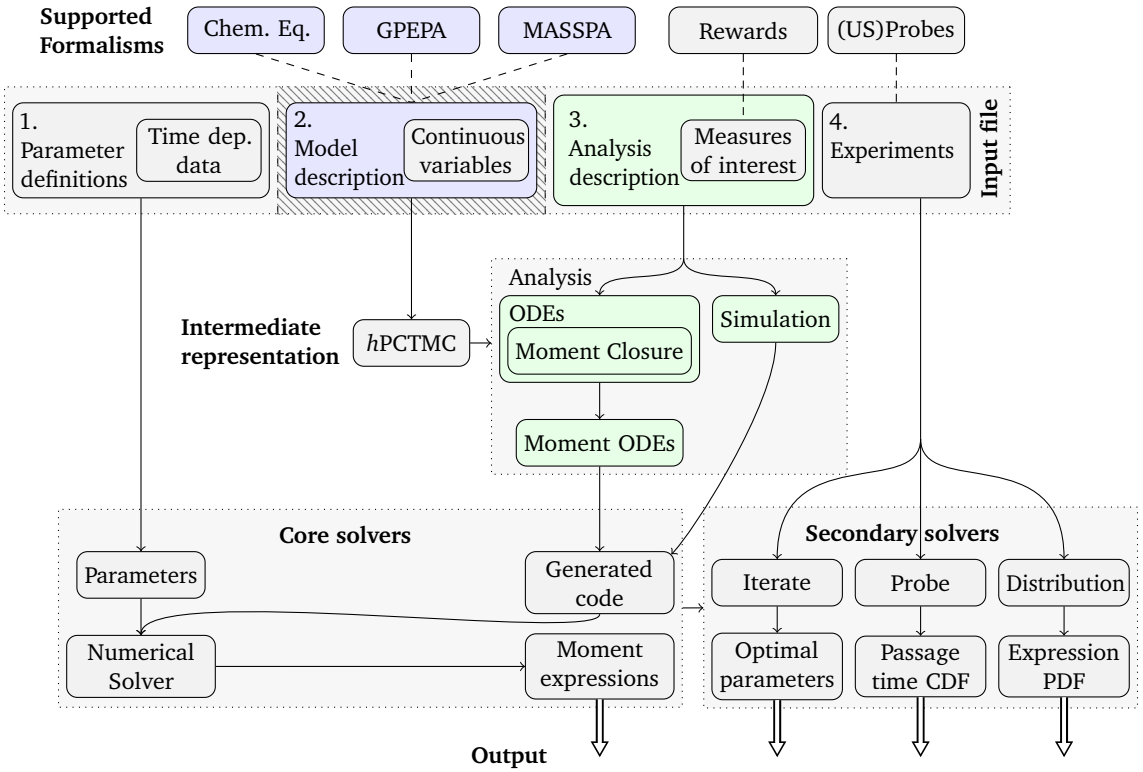


Figure 8.1: Overview of the architecture of GPA.

```
rreq   = 2.0;   rdata  = 1.0;   rthink = 0.2;
rbreak = 0.1;   rreset = 2.0;
n_C = 100;   n_S = 50;
```

Time-dependent rates, such as the rate $\lambda(t)$ in the example in Section 6.6.2 can be loaded from a CSV file, where the first column specifies the time of change of the rate and the given column (2 in the code bellow) the value of the rate:

```
load "worldcup.csv" 2 into lambda;
```

Each PCTMC specification language supported by GPA has to provide syntax for model definition and syntax for referring to population labels.

8.2.1 Plain PCTMC

Models can be specified in a syntax close to the chemical-reaction style notation in Equation 3.1. Each population label is written in curly braces, such as `{Name}`. Rates can be written as arithmetic expressions over population labels. After a number of transition class definitions separated by semi-colons, the initial values of non-zero populations are given as `{Name} = n`. For example, the peer-to-peer model from Section 3.2.1 is:

```
{Soff} -> {Son}  @ {Soff} * ron;   {Ul} + {Us}  -> {Us} + {Us}  @ {Us}{Ul} * rseed;
{Son}  -> {Soff} @ {Son} * roff;   {Son} + {Ul} -> {Son} + {Us} @ {Ul}{Son} * rsseed;
{Us}   -> {Uf}   @ {Us} * rleave;  {Son} + {Uf} -> {Son} + {Uf} @ {Uf}{Son} * rsseed;
{Uf}   -> {Ul}   @ {Uf} * rback;

{Soff} = n_S;
{Ul}   = n_Ul;
{Us}   = n_Us;
```

8.2.2 GPEPA

GPA implements a large subset of GPEPA syntax, described in Section 2.2.3. A model definition consists of a number of sequential agent definitions and a system equation. For example the client–server model from Section 3.2.3 can be defined as:

```
Client          = (req, rreq).Client_waiting;
 Client_waiting = (data, rdata).Client_think;
 Client_think   = (think, rthink).Client;

Server            = (req, rreq).Server_get + (break, rbreak).Server_broken;
   Server_get     = (data, rdata).Server;
   Server_broken  = (reset, rreset).Server;

Clients{Client[n_c]}<req,data>Servers{Server[n_s]}

Count think;
```

The last line specifies an action counting population in the resulting PCTMC that keeps track of the count of fired *think* actions and can be used as an impulse reward specification, as in Section 5.2.2. Agent populations are referred to by `Group:Agent` syntax and action counting populations as `#action`.

The probed version of the client–server model can be defined by including an auxiliary sequential agent that remembers a fired *think* action and modifying the system equation to include a client synchronised with such an agent, as described in Section 3.5.2:

```
...
NotDone = (think, rthink).Done;
Done    = (think, rthink).Done;
...
Clients{Client[n_c-1]|Client<think>NotDone}<req,data>Servers{Server[n_s]}
```

8.2.3 Spatial process algebra

Additional model definition languages can be easily added to GPA. The code below shows an example of a model of a content distribution network described in the MASSPA process algebra:

```
Client_stale@x   = ?(m, 0.25).Client_updated@x + ?(m, 0.75).Client_stale@x;
Client_updated@x = (r_stale@x).Client_stale@x;

Server@x         = !(r_rx, m, 0.1).Server@x + (r_stale@x).Server_refresh@x;
Server_refresh@x = ?(m, 0.1).Server@x + ?(m, 0.9).Server_refresh@x;

Server@A[N_A]<>Server_refresh@B[N_B] <> Server_refresh@C[N_C]<>Client_stale@D[N_D]
<> Client_stale@E[N_E]<>Client_stale@F[N_F];

Channel(Server@A,Server_refresh@B,m,2); Channel(Server@A,Server_refresh@C,m,1);
Channel(Server@B,Client_stale@D,m,3); Channel(Server@B,Client_stale@E,m,1);
Channel(Server@C,Client_stale@E,m,2); Channel(Server@C,Client_stale@F,m,2);
```

Further details of the MASSPA process algebra can be found in the paper by Guenther and Bradley [92] and the PCTMC semantics in the paper by Bradley et al. [2].

8.2.4 *h*PCTMC continuous variables

A model definition can be extended with continuous variables, thus defining a *h*PCTMC model. Each continuous variable is defined by its accumulation function and an initial value. For example, the temperature variable $\mathcal{T}$ from Equation 6.2 can be defined as:

```
ddt ~T = {S1} * rheat * c / v - {A1} * rcool * c / v;
    ~T = 0;
```

Continuous variables can be used within model definitions, such as in the on-rate of air conditioning units in the example in Section 6.2:

```
...
{A0} -> {A1} @ {A0} * min(tthresh - ~T, 0.0) * (-r_on);
...
```

Accumulated populations and products of populations are a special case of continuous variables and don't require the `ddt` definition. We use the notation `acc(h)` for the accumulated product of populations given by `h`.

8.2.5 Variables and pattern matching

The solvers below allow computation of moments of populations. For convenience, GPA additionally allows definition of variables which can be arithmetic expressions over populations. For example, the energy and total reward in the client–server model, defined in Equation 5.1 and Equation 5.2, can be specified as:

```
$energy = crunning * acc(Servers:Server_get)
  + cpower * acc(Servers:Server) + cbroken * acc(Servers:Server_broken);

$total = cfee * #task
  + (-crunning) * acc(Servers:Server_get) + (-cpower) * acc(Servers:Server)
  + (-cbroken) * acc(Servers:Server_broken);
```

In case of GPEPA models, GPA also provides a convenient shorthand for sums of populations of sequential PEPA agents based on their structure. The pattern `%Group:_<actions>Agent` expands to the sum of all populations where `_` can be replaced by any agent. For example, to sum over the whole absorbing set of client states in the probed client–server example, Equation 3.18, we can use the following syntax:

```
$passage = %Clients:_<task>Done;
```

8.3 Model analysis – Core solvers

The model description is followed by a list of solvers applied to the model. Core solvers compute transient evolution of moment based expressions in the model. Each solver accepts pre-processing options (in square brackets) that influence the type of analysis to be performed. These options, together with the model and desired expressions (in curly braces) generate a symbolic representation of the solver. This representation is compiled into a numerical solver, with options given in parenthesis. The `stopTime` option gives the time t_f until which the numerical solution is obtained and `stepSize` determines the fixed time step at which the expressions are sampled. The code below shows the general structure:

```
Solver[option = value, ...]
  (stopTime = t_f, stepSize=1, numerical_option = value, ... ) {
   moment_expression, ... -> output file;
   moment_expression, ... -> output file;
  ...
}
```

Moment expressions

Each core solver can plot a number of given expressions evaluated. These can be any expressions that can be evaluated from the raw moments represented by the ODEs in case of ODE analysis. An expectation of a population is denoted by `E[X]` and higher-order moments similarly by `E[X Y Z]` etc. GPA provides convenient shorthands such as `Var[X]` for variance of a population `X`, `Cov[X,Y]` for the covariance of two populations `X` and `Y`, `SCM[X,n]` for the n-th standardised central moment of `X`. Additionally, these shorthands are defined on expressions that are linear combinations of populations and GPA automatically expands the resulting expectation into a corresponding linear combination of raw moments, for example `Var[a*X + b*Y]` is translated to the corresponding expression involving raw moments of populations. The n-th raw moment of an expression is denoted by `Moment[e, n]`.

8.3.1 ODE analysis

ODE analysis of moments is implemented in the `ODEs` solver as shown in the code below:

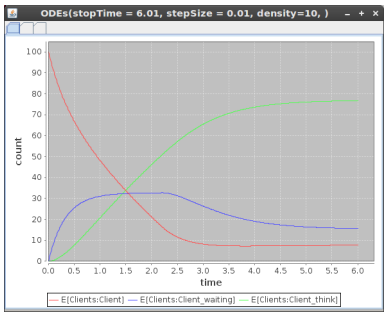
```
ODEs[momentClosure=..., maxOrder=...]
     (stopTime=..., stepSize=..., integrator=...){ ... }
```

Pre-processing options determine the used moment closure and the maximum order of moments in the system. When no option is specified, the mean-field closure from Section 3.4.1 is applied; value `NormalClosure` selects the normal closure for polynomial rates from Section 3.4.3 and `NormalClosureMinApproximation` the min-normal closure from Section 4.4. The resulting system of ODEs is numerically integrated until the time given by `stopTime`. A number of algorithms is available, via the `integrator` parameter – `ClassicalRungeKutta`, `DormandPrince853` and others, provided by the *Apache Commons Math* library [61].

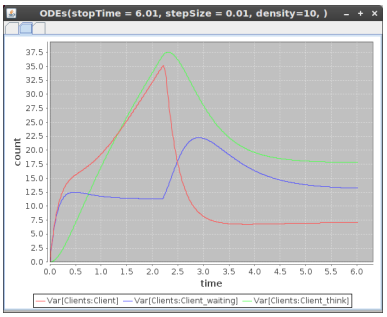
Figure 8.2 shows an example of the ODE solver applied to the GPEPA client–server example. Figure 8.3 shows additional functions that can be used in the moment based expressions, such as the upper and lower moment based bound on the CDF of a population, using the technique of Tari et al. [173].

```
ODEs(stopTime=6.0, stepSize=0.01, density=1){
    E[Clients:Client],
       E[Clients:Client_waiting],
       E[Clients:Client_think];//          (b)
    Var[Clients:Client],
       Var[Clients:Client_waiting],
       Var[Clients:Client_think];//        (c)
    E[$switch],
       E[$switch]+2.58*Var[$switch]^0.5,
       E[$switch]-2.58*Var[$switch]^0.5;// (d)
}
```

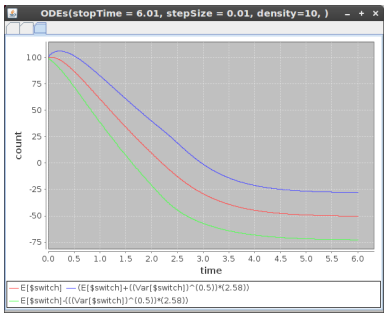
(a) Code



(b) Client means



(c) Client variances



(d) Switch point distance

Figure 8.2: Example plots from the ODE analysis of the client–server model, corresponding to plots from Figure 4.2 and Figure 4.1a.

8.3.2 Simulation

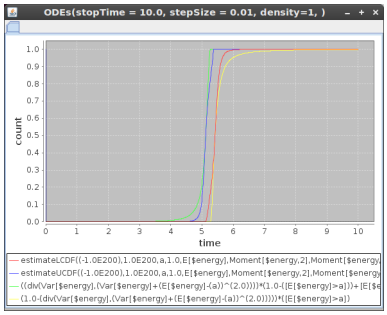
The `Simulation` solver implements the Gillespie algorithm for simulation of PCTMC models, described in Section 3.3:

```
Simulation(stopTime = ..., stepSize = ..., replications = ...) { ... }

AccurateSimulation(stopTime = ..., stepSize = ...,  CI = ...
                   maxRelCIWidth = ..., batchSize = ...) { ... }
```

```
ODEs(stopTime=10.0, stepSize=0.01, density=10){
estimateLCDF(0,1e200,a,1, E[$energy],
  Moment[$energy, 2], Moment[$energy, 3],
  Moment[$energy, 4], Moment[$energy, 5],
  Moment[$energy, 6]),
estimateUCDF(0,1e200,a,1, E[$energy],
  Moment[$energy, 2], Moment[$energy, 3],
  Moment[$energy, 4], Moment[$energy, 5],
  Moment[$energy, 6]),
div(Var[$energy], Var[$energy]+(E[$energy]-a)^2)
  *(1-[E[$energy] > a]) +[E[$energy] > a],
(1-div(Var[$energy], Var[$energy]
  +(E[$energy]-a)^2)) *[E[$energy] > a];}
```

(a) Code



(b) Screenshot

Figure 8.3: Example plot of reward completion time bounds expression, corresponding to Figure 5.6.

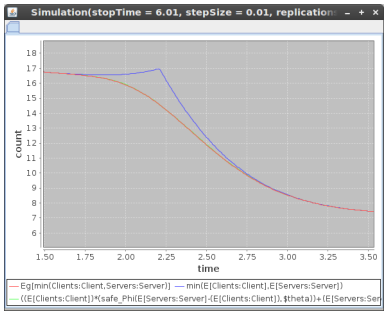
The parameter `replications` specfies the number of simulation traces that are used to compute estimates of the given moment based expressions. Alternatively, the version `AccurateSimulation` accepts a confidence level `CI` with a desired relative width `maxRelCIWidth` and a parameter `batchSize` that determines how frequently the analysis estimates each confidence interval.

In addition to moment based expressions, the analysis can compute an expectation of an arbitrary expression, with the syntax `Eg[...]`. Figure 8.4 shows an example, corresponding to a plot in the investigation of the min-normal closure in Section 4.4.

```
$theta = (Var[Clients:Client] - 2*(E[Clients:Client Servers:Server] -
 E[Clients:Client]*E[Servers:Server]) + Var[Servers:Server])^0.5;
```

```
Simulation(stopTime=6.0, stepSize=0.1,
           replications=100000){
Eg[min(Clients:Client, Servers:Server)],
min(E[Clients:Client], E[Servers:Server]),
E[Clients:Client] * safe_Phi(
  E[Servers:Server] - E[Clients:Client], $theta)
 + E[Servers:Server] * safe_Phi(
  - E[Servers:Server] + [Clients:Client], $theta)
 + $theta * safe_phi(
  E[Servers:Server] - E[Clients:Client], $theta);
}
```

(a) Code



(b) Screenshot

Figure 8.4: Example plot of simulation estimates, corresponding to Figure 4.10. Functions `safe_Phi(a,b)` and `safe_phi(a,b)` stand for $\Phi(a/b)$ and $\phi(a/b)$ respectively, with correctly handling the positive and negative infinity in case b is zero.

8.4 Experiments – Secondary solvers

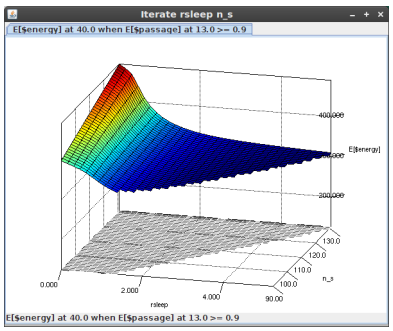
8.4.1 Parameter exploration

The provided ODE and Simulation analyses can compute various performance and energy metrics for models under a single parameter regime. GPA also provides further capabilities which allow to analyse models under a number of parameter values and also embed the ODE or simulation analysis in a global optimisation problem.

The `Iterate` experiment accepts a number of parameter range specifications of the form `r from 0.0 to 1.0 in 10 steps`. For each combination of parameter values from the given ranges, the experiment performs a given analysis. If only one or two ranges are given, at each parameter combination, the experiment can show a 2d or 3d plot respectively of an expression evaluated at given points in time. Additionally, it is possible to specify an inequality constraint which has to hold for the value to be recorded/shown, for example representing satisfaction of an SLA. Figure 8.5 shows an example from the performance–energy trade-off in Section 5.5.

```
Iterate rsleep from 0.0 to 6.0 in 40 steps
        n_s from 90.0 to 131.0 with step 1.0
  ODEs(stopTime=40.1,stepSize=0.1,density = 10)
  plot{
    E[$energy] at 40.0
      when E[$passage] at 13.0 >= 0.9;
}
```

(a) Code



(b) Screenshot

Figure 8.5: Iterate experiment, corresponding to Figure 5.10a.

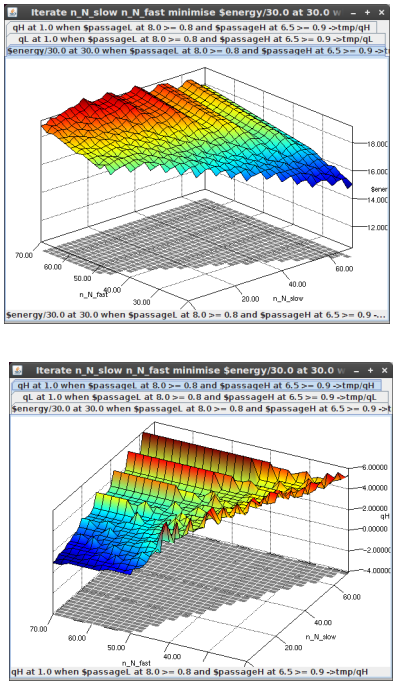
Parameters can be also explored in an optimisation problem, where the objective is to minimise a given moment based expression evaluated at a given time, under a number of inequality conditions. The `Minimise` experiment accepts an objective expression and inequality constraints, number of parameter range specifications and a list of plot expressions as the `Iterate` experiment. For each plot expression, the experiment outputs the value evaluated with parameters that optimise the objective and satisfy the constraints. Additionally, the `Minimise` experiment can be a part of an `Iterate` experiment, where the minimisation occurs at each resulting parameter combination. Figure 8.6 shows an example from optimising a cluster scheduling policy in Section 7.4.4.

8.4.2 Unified Stochastic Probes for GPEPA

GPA supports passage time specifications with Unified Stochastic Probes [103] through the `Probe` solver. Figure 8.7 shows an example of an individual transient passage time probe. More details of the implementation can be found in Kohut [124].

```
Iterate
  n_N_slow from 0.0 to 70.0 in 36 steps
  n_N_fast from 20.0 to 70.0 in 26 steps
    Minimise E[~Aenergy] at 30.0
               when E[$passageL] at 8.0 >= 0.8
                and E[$passageH] at 6.5 >= 0.9
     qL from 0.0 to 3.0 in 20 steps
     qH from 0.0 to 6.0 in 20 steps where
         r_assign_L_slow_0=(3.0-qL)*2/3;
         r_assign_L_slow_1=(3.0-qL)*1/3;
         r_assign_L_fast_0=qL*2/3;
         r_assign_L_fast_1=qL*1/3;
         r_assign_H_slow_0=(6.0-qH)*2/3;
         r_assign_H_slow_1=(6.0-qH)*1/3;
         r_assign_H_fast_0=qH*2/3;
         r_assign_H_fast_1=qH*1/3;
  ODEs(stopTime = 30.1, stepSize = 0.1,
        density = 10)
     plot { E[~Aenergy] at 30.0
          when E[$passageL] at 8.0 >= 0.8
          and E[$passageH] at 6.5 >= 0.9;
          qH at 1.0
          when E[$passageL] at 8.0 >= 0.8
          and E[$passageH] at 6.5 >= 0.9;
     }
```

(a) Code



(b) Screenshot

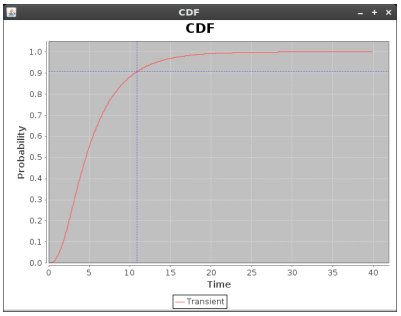
Figure 8.6: Parameter exploration with minimisation at each combination, corresponding to Figure 7.5c (top screenshot) and the optimal value of the `qH` parameter (bottom screenshot).

8.4.3 Distribution computation

The primary simulation solver only computes expectations and moment based expressions. GPA provides a secondary solver that can also compute distributions of arbitrary expressions involving populations and continuous variables. For example, Figure 8.8 shows the distribution of the temperature variable in the air-conditioning model in Section 6.2.

```
Probe
   ODEs(stopTime = 40.0,stepSize = 0.1,density = 10)
     transient 300 {
      GProbe = begin: start, end:stop
      observes { LProbe = eE: begin, think: end }
      where {
         Clients{Client[n_C]}
           => Clients{Client<*>Probe|Client[n_C-1]}
      }
   }
```

(a) Code

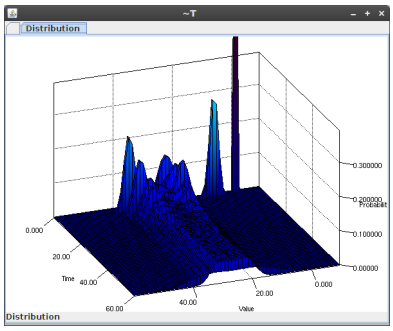


(b) Screenshot

Figure 8.7: Example of a local probe in the client–server model.

```
Distribution
  Simulation
    (stopTime=61.0, stepSize=1.0,
     replications=100000)
  computes {
      $temp into 60 bins;
  }
```

(a) Code



(b) Screenshot

Figure 8.8: Distribution of the temperature variable in the *h*PCTMC client–server model with air-conditioning corresponding to Figure 6.9.

8.5 Implementation details

GPA is implemented in the Java programming language and all the source code is freely available on the tool website [87] under the MIT license [177]. We highlight a few implementation features:

Modular parser Parsing of input files is implemented with the *ANTLR 3* parser [20]. A core parser implements all the syntax except for model definition and population labels. Each specific language parser is responsible for defining these rules and for implementing a translation of models to PCTMCs. For example, the GPEPA parser defines the syntax of GPEPA models, closely following the formal definition. It also provides a translation to PCTMCs, impelenting the semantics in Section 3.2.2. Population labels are defined as group–component pairs.

Symbolic expression library Arithmetic expressions in GPA are based on a custom built symbolic expression library. These include transition class rates, moment and population based expressions in solvers and generated ODE methods. The library provides tools for convenient implementation of expression transformations, used for example in the implementation of the various moment closures.

Abstract solver representation Each solver is represented in a symbolic abstract form. For example, the ODE analysis generates a corresponding symbolic ODE method. Before the numerical solution stage, this form is printed into a target language code and dynamically compiled into concrete implementation. GPA additionally optimises the generated code, for example by factoring out common arithmetic expressions into temporary variables. By default, the target language is Java and the code is compiled in memory. The compiled method is provided to one of the built-in ODE solvers. The solver supplies a complete parameter set to the ODE method and calls the method in each step of the numerical integration.

Solution transformation Each solver (ODE or stochastic simulator) outputs a vector for each selected time instant containing values of all computed moments in the PCTMC. The symbolic expression library is used dynamically compile user supplied expressions in order to efficiently transform the raw moment results into the desired form.

Parallelisation In order to speed up the parameter exploration experiments, GPA is able to distribute the computation of the numerical solutions for different parameters onto a given number of CPU cores. Moreover, GPA can also automatically split the computation into a large number of smaller tasks and generate a specification used by the *HTCondor* high-throughput distributed computation framework [176]. This allowed us for example to evaluate the error in Figure 6.13 by running the simulations on around 100 workstations.

8.6 Conclusion

In this chapter, we presented GPA, a software tool which implements most of the techniques developed in this thesis. The architecture of GPA closely follows the structure of the theoretical framework. At the core of GPA is a PCTMC representation, which serves as an intermediate format for the models. Different specification languages are supported in a separate layer, only requiring an implementation of the corresponding PCTMC semantics. GPA can be used by practitioners to apply the framework of this thesis to real models. Additionally, the source of GPA is available under an open source licence and the architecture facilitates extensions of the modelling formalisms and solution techniques.

Chapter 9

Conclusion

9.1 Summary of achievements

This thesis has explored methods for efficient analysis of performance–energy trade-offs in large-scale Markov population models. As a motivation, we aim to be able to analyse the trade-off in models of large data centres. In the introduction to this thesis, we argue that the resulting approach has to be able to

(i) analyse models with very **large state space**,

(ii) provide **high-level behavioural model descriptions**,

(iii) capture detailed **SLA specifications** based on passage time probabilities,

(iv) capture **energy consumption metrics**,

(v) jointly capture the interaction between **temperature and workload** in the system,

(vi) include **time-dependent** workloads.

To the best of our knowledge, no existing prior work was able to simultaneously meet all these requirements. Analysis of such models was either restricted to significantly smaller state spaces, or the available metrics were limited to average and often steady-state measures. Time-dependent behaviour or feedback from continuous quantities such as temperature was not possible. The main contribution of this thesis is a framework where all these restrictions are lifted for a class of Markov population models. In achieving this, we are now able to rapidly analyse performance–energy trade-offs in realistic models of large-scale systems.

Our approach is based on efficient analysis of Markov population models where the system is approximated with a set of ordinary differential equations (ODEs). These include the *mean-field* method [42, 31, 33], *moment closure* heuristics [186, 71, 132, 81, 99, 17] and *fluid analysis* of process algebra models [108, 183, 68, 56]. These techniques can rapidly deal with extremely large state spaces and are not sensitive to the number of different agents in the system.

As the first step, we defined *Population Continuous-Time Markov Chains* (PCTMC) in Chapter 3, a stochastic process to which most of the ODE-based techniques apply. In particular, fluid analysis of stochastic process algebras, such as the method of Hillston [108], Hayden and Bradley [99] or Bortolussi and Policriti [41] can be considered as an application of a moment-closure heuristic to PCTMCs derived from a process algebra model. An advantage of the method of Hayden

and Bradley [99] is that it provides an extension [4] which can use the ODE analysis to obtain passage-time probabilities in the models and thus capture the required SLA specifications. We generalised this approach to PCTMCs in Chapter 3. Before proceeding with further extensions, we investigated the accuracy of different ODE analysis techniques. In Chapter 4 we presented a novel moment closure for models with rates containing the $\min$ function.

In Chapter 5, we extended the ODE analysis of PCTMCs to capture moments of *accumulated rewards*. Our extension supports both rate and impulse rewards and is suitable for modelling energy consumption and other more complicated cost functions. Moreover, both rewards and passage time probabilities can be computed in the same analysis. This allowed us to define the energy–performance trade-off as a constrained global optimisation problem with an embedded system of differential equations. Although no known algorithms can provide guaranteed solutions to the general class of such problems, the low computational cost of ODE analysis means that we were able to apply approximate numerical algorithms, including explicit parameter exploration.

To complete the analysis related requirements set above, we defined *hybrid PCTMCs* (*h*PCTMC) in Chapter 6. A *h*PCTMC is a stochastic process that consists of a PCTMC, augmented with continuous variables, defined by integral equations over the populations and other continuous variables in the model. To introduce feedback loops into *h*PCTMC models, the continuous variables are allowed to influence rates in the PCTMC part of the model. For example a continuous variable can represent temperature that affects behaviour of air-conditioning units in a data centre. We extend the ODE analysis to *h*PCTMC. Importantly, by treating time as such a continuous variable, this allows us to capture time-dependent rates in *h*PCTMC models.

In Chapter 7, we defined GPEPAc, an extension of GPEPA where *channels* can serve as building blocks of complex interactions, which can be used to express session-based communication protocols. GPEPAc addresses the limitation of existing specification languages which are amenable to ODE analysis. These restrict interactions between agents in the system to single-step transitions. To be able to analyse models in this formalism, we defined a corresponding PCTMC semantics. We illustrated the language on a model of a heterogeneous computing cluster and used rapid ODE analysis to capture energy–performance trade-off under multiple SLAs.

We kept an up-to-date implementation of our framework in a software tool GPA. We gave a brief overview of GPA in Chapter 8. GPA allowed us to perform a range of case studies and experiment with different heuristics used by the ODE analysis. Moreover, the architecture of GPA closely resembles our theoretical approach and keeps PCTMC as an intermediate representation of models. Different specification languages can be included in a lightweight separate layer. For example, *MASSPA*, a spatial process algebra, is implemented by a corresponding translation to PCTMC [92].

9.2 Applications

In this section, we demonstrate the complete analysis life-cycle provided by the framework developed in this thesis. We show how *h*PCTMC models can be used by the provider of a *distributed high-throughput cycle-stealing system*. Such systems, for example BOINC or HTCondor [134],

are a popular computation model amongst scientists, engineers and financial organisations, allowing massively distributed calculations to be spread over many thousands of otherwise idle office workstations. We apply our framework to real data from a HTCondor cluster deployed at Newcastle University [1].

9.2.1 Distributed high-throughput cycle-stealing system

In total, around 1400 machines take part in the cluster. These are workstations from various classrooms and spaces around the university campus. There is a large variety of usage patterns – some machines are used for teaching and are physically inaccessible outside teaching hours, some belong to 24 hour access facilities. Some of the machines are located in halls of residence and experience higher use during weekends.

HTCondor monitors each workstation for any interactive user activity. If a machine is idle for a period of time, HTCondor marks it as available for high-throughput computation. The cluster is accessible to a number of high-throughput user groups. Such users submit their computational tasks, usually parallelised into a large number of separate jobs, into one of several queues for the different groups. HTCondor continuously monitors each queue and assigns jobs to available workstations. If a user arrives at a workstation executing a high-throughput job, HTCondor interrupts the execution of the job and *evicts* it back to the queue. A high-level overview of the system can be seen in Figure 9.1.

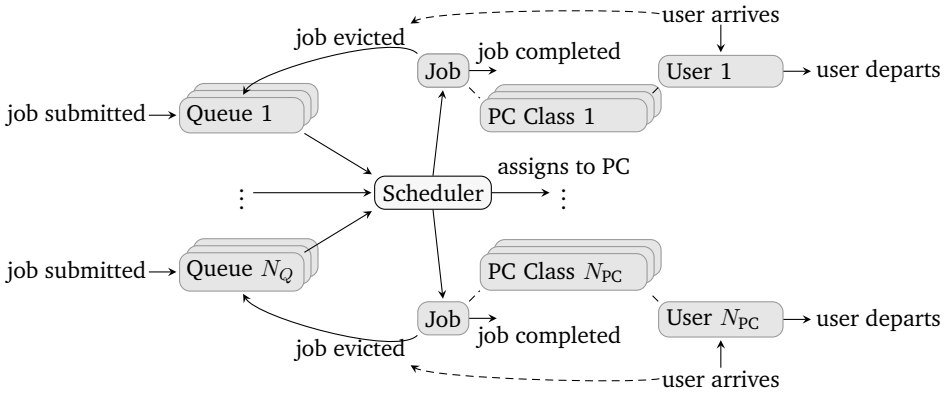


Figure 9.1: Overview of the model of HTCondor.

9.2.2 GPEPAc model of HTCondor

We describe a model of the system in the GPEPAc process algebra developed in Chapter 7. The session-based communication features of GPEPAc are particularly suitable for this application. In this case, we use multi-phase synchronisation to implement job durations with Coxian distribution [e.g. 23]. Appendix E shows a detailed GPEPAc model of the system. Here, we only list the key points of this model:

(i) The model contains agents for users, jobs, workstations and HTCondor schedulers.

(ii) The job processing durations are distributed according to Coxian random variables. This is achieved by a number of states of the respective agents.

(iii) GPEPAc channels represent allocation of jobs and users to workstations. The channel connection between a job and a workstation is maintained throughout all the phases of the respective Coxian distribution.

In this example, we use a simple instance of this model consisting of only one type of workstations and users – A for "all" users and workstations from the campus. We consider two job groups – B for the "background" jobs submitted by all the different high-throughput groups, and H for "hypothetical" jobs that we will use to evaluate the system under various conditions.

We use this model to showcase the many developments that this thesis provides in the analysis of energy and performance in systems with time varying workloads:

(i) The resulting GPEPAc model can be automatically translated to a PCTMC, using the semantics described in Section 7.3.

(ii) We augment this PCTMC model by adding a continuous variable $\mathcal{Y}_{energy}(t)$ capturing the energy consumption caused by executing the hypothetical H jobs and thus obtain a *h*PCTMC model, as described in Chapter 6.

(iii) Using the method in Section 5.2.2, we can define an additional population $X_{evict}(t)$ counting the number of evictions of the hypothetical jobs.

(iv) The efficient ODE analysis described in Section 6.3 can be used to rapidly obtain means and higher moments of $\mathcal{Y}_{energy}(t)$ and $X_{evict}(t)$.

(v) The method from Section 3.5 can be used in the same ODE analysis to compute passage time distribution of the time T_H that each job from a group H takes to finish.

(vi) The model is analysed with the GPA tool described in Chapter 8.

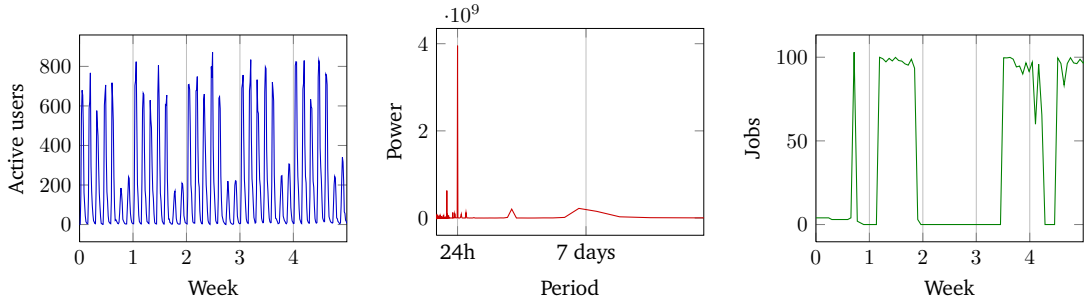
(vii) The tool uses techniques from Section 6.6 to accept data driven time-dependent rates, such as the user arrival rate $r_{arrive,A}(t)$.

9.2.3 Analysis using the collected data

HTCondor keeps a log of all system events, such as job submissions and user arrivals. We were able to obtain a complete log for the year 2010. For the purposes of the GPEPAc model, we aggregated the log to only keep timestamps with the information about increasing and decreasing populations. Figure 9.2 shows the number of active users and the number of currently processed jobs (both submitted and executing) in the entire system, throughout a sample period of five weeks (during October 2010). It also shows a periodogram for the number of active users. We demonstrate two different approaches to using HTCondor logs together with the GPEPAc model:

Static model: In the static model, we build a version that captures the HTCondor system during a longer time period. This model can be used to evaluate different system configurations and their effects on the high-throughput performance and overall energy consumption.

Dynamic model: The dynamic model runs alongside the real system and uses the logs to obtain current model parameters. The model can be used to evaluate short term hypothetical scenarios, such as the effect of submitting a batch of jobs.

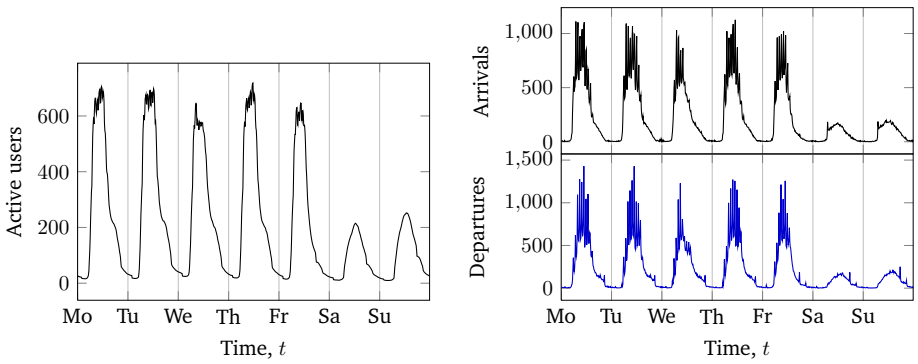


(a) Active users (b) Periodogram for active users (c) Jobs

Figure 9.2: Sample trace of the number of active users during five weeks of monitoring the system.

Static model of an average week

We can see from Figure 9.2 that the number of active users in the system almost repeats with a weekly cycle. This holds for the rest of the data (excluding weeks outside term times) and therefore it is possible to construct a model which captures a "representative" week in the system. We take one such week and fit the time-dependent user arrival and departure rates $r_{arrive,A}(t)$ and $r_{logout,A}(t)$. In this example we simply fit an exponential distribution with a mean equal to the average inter-arrival time throughout each one hour interval during the week. Figure 9.3 shows the number of users during a representative week and the respective rates.



(a) Active users (b) Arrival and departure rates

Figure 9.3: Number of active users during a representative week and the corresponding arrival and departure rates (per hour).

We perform a similar fit to the submission rates and durations of background jobs, rates $r_{submit,B}(t)$ and $r_{process,B}$. In case any of the durations require a Coxian distribution, experiments show that the expectation maximisation algorithm of the *EMpht* tool [22] gives good match to both mean and variance of the distributions.

We will use the model to evaluate the performance of the system as seen by a batch of 100 hypothetical jobs, each with a two-phase Coxian duration with mean 2.7 hours. For example, we

can compare the performance of the system if the jobs were submitted at different times of the week. We choose a busy time such as 12pm on Monday and a less busy time, such as 6pm on Friday. Figure 9.4 shows the passage time probabilities for a single job to finish if the batch was submitted at these two times, computed as described in Section 6.6.1.

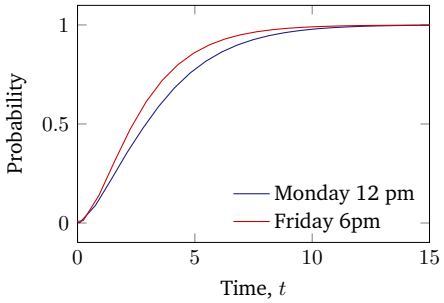
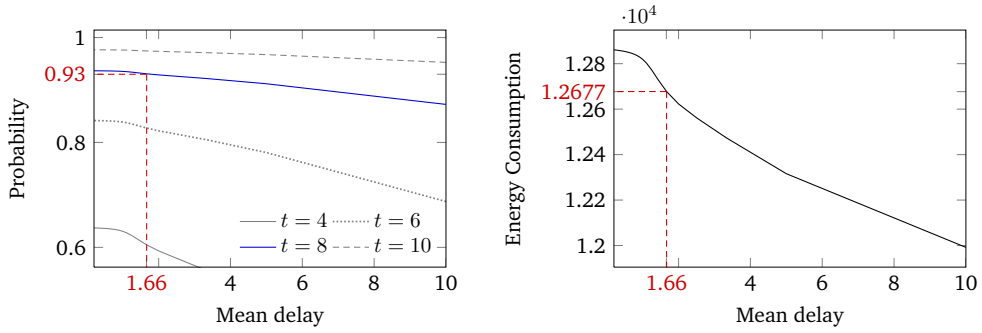


Figure 9.4: Probability of a single hypothetical job finishing after being submitted at different times.

One parameter that the maintainer of HTCondor can control is the delay after which workstations become available to high-throughput jobs. We can set a hypothetical SLA of an individual job finishing within 8 hours at least 93% of the time. Our aim is to minimise the energy consumption caused by the high-throughput jobs. If the availability delay is too short, jobs get allocated to workstations during busy times and are likely to be evicted. On the other hand, a longer delay avoids job evictions but potentially increases the time each job spends in the system. We can use the rapid ODE analysis of the model to evaluate the performance and energy metrics for a large number of delay parameters. Figure 9.5 shows that an average availability delay of 1.66 hours minimises the total energy consumption of the batch of jobs and satisfies the given SLA.



(a) Probability of a single job finishing before time t

(b) Energy consumption after finishing all jobs

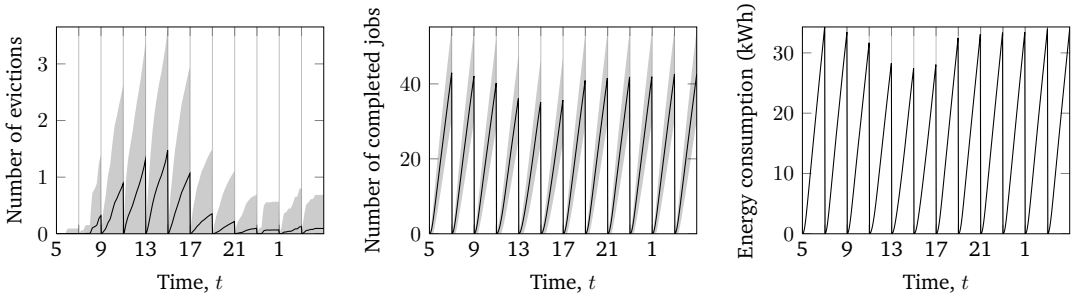
Figure 9.5: Energy–performance trade-off under varying mean availability delay. The SLA is given by the probability at $t = 8$ to be at least 0.93. Probabilities for other values of t are shown for illustration.

Dynamic model

Another potential use of the GPEPAc model of the HTCondor system is to evaluate short-term predictions on an accurate model of the current state of the system. The aggregate nature of PCTMCs makes it feasible to process the logs of HTCondor in an on-line fashion to obtain current parameters of the system. At each point in time, we forecast user and background job arrival parameters to fit a *dynamic model* for a subsequent *forecast period*. In this example we set this

period to be two hours and we use a simple forecast by looking at the parameters from the same time during the previous week.

We can use the dynamic model to evaluate derived performance metrics of the system. For example, we can use the model to virtually submit a hypothetical job that counts the number of evictions a job is likely to experience. Figure 9.6a shows this for the different forecast periods. Another virtual experiment might be to evaluate the submission of a batch of 500 jobs with mean processing time 2 hours and evaluate the number of completions at the end of the forecast period, as seen in Figure 9.6b. Figure 9.6c shows the resulting additional energy consumption caused by these jobs. The rapid ODE analysis allows calculation of a large number of such experiments. For example, the analysis could be used by users of HTCondor to predict the time of completion of their submitted jobs.



(a) Evictions (b) Completions (c) Additional energy consumption

Figure 9.6: Forecast for metrics of the hypothetical batch of jobs.

9.3 Future work

There are many potential avenues for future work. Here we choose to highlight, in our opinion, six of the most impactful and promising enhancements to the energy–performance framework presented in this thesis.

9.3.1 Practical advances

A number of enhancements is motivated by applications, such as the model of HTCondor in this section, that in our opinion would have the greatest effect on the performance modelling accuracy of the framework.

General distributions In PCTMC models, all transitions are assumed to be exponentially distributed. We have suggested how to represent Coxian distributions in the GPEPAc model of HTCondor in Section 9.2. These can approximate a range of realistic distributions with a sequence of exponentially distributed phases. However, certain distributions are notoriously hard to approximate in this way. An example includes deterministic distributions, where the system takes an action after a pre-determined time period. For example, the availability delay in the model of HTCondor is usually a deterministic delay. There has been an ongoing

work into incorporating deterministic delays and general distributions into ODE analysis of Markov population models [95, 96, 39]. So far, existing results apply only to a limited number of cases.

More accurate parameter forecasts The dynamic model of HTCondor from Section 9.2 relied on accurate parameter forecasts from existing data. We are investigating suitable time-series analysis techniques which would allow us to obtain accurate predictions of time-dependent rate parameters. In particular, we seek methods which would be able to forecast parameters of Coxian distributions in the model.

Different synchronisation options in GPEPAc transactions In the presented version of GPEPAc, agents within a transaction can only take part in one type of synchronisation, where a single sender emits a message on a set of channels to a set of receivers. Various applications might require different types of synchronisation. For example, agents might be able to probe a socket connection and only perform an action if there were enough required receiving agents.

Support of dynamic models in GPA We are planning to implement the full modelling life-cycle presented on the HTCondor study in GPA. This involves on-line aggregation of system logs, fitting and forecasting model parameters and automatically generating models for different types of users.

9.3.2 Theoretical advances

We present two example theoretical developments that, if implemented, would greatly enhance the modelling power of the presented framework.

Accumulations and rates with thresholds When describing *h*PCTMC models in Chapter 6, we used the minimum function in the transition rate of an air conditioning unit to capture the feedback from the temperature variable. We were able to use the min-closure from Section 4.4 to obtain accurate approximations from the ODE analysis. Often, it is natural for the model to implement feedback based on *thresholds*, such as enabling a transition only if the temperature variable is above a certain fixed threshold. Such a rate might introduce discontinuities in the ODE system. In order to achieve this, we would need to develop a specialised moment-closure that would express expectations involving such terms as functions of other moments.

Hybrid state space of individual agents The continuous variables in *h*PCTMC are available globally to all agents. However, often the behaviour of individual components of large-scale systems, such as wireless sensor networks, is influenced by their "local" continuous variables such as battery level or position in space. There has been preliminary work into extending Markov population models by allowing each of the large number of agents to access a collection of local continuous variables [97]. A natural extension of the analysis is a system of partial differential equations (PDEs), that can compute the proportion of agents which have a particular value of local variable. So far, this approach has been limited to only a small class of Markov population models. Moreover, we believe that a similar approach

can be used to address the efficiency of ODE analysis of GPEPAc models. The session-based interactions in GPEPAc models can lead to a combinatorial explosion of the number of different transactions in the system. For example, in the model in Section 7.4, the ODE analysis would become infeasible if each of the nodes is allowed to concurrently serve 100 different jobs. However, we could approximate the number of each different job as a local variable and use the PDE method to obtain an approximation to the system.

Related Publications

[1] J. T. Bradley, M. Forshaw, A. Stefanek and N. Thomas. “Time-inhomogeneous population models of a cycle-stealing distributed system”. In: *UKPEW’13, The 29th UK Performance Engineering Workshop*. 2013 (on p. 21, 170).

[2] J. T. Bradley, M. C. Guenther, R. A. Hayden and A. Stefanek. “GPA - A multiformalism, multisolution approach to efficient analysis of large scale population models”. In: *Theory and Application of Multi-Formalism Modeling*. Ed. by M. Gribaudo and M. Iaconno. IGI Global, 2013. ISBN: 1466646594. DOI: 10.4018/978-1-4666-4659-9 (on p. 19, 156, 160).

[3] M. C. Guenther, A. Stefanek and J. T. Bradley. “Moment closures for performance models with highly non-linear rates”. In: *Computer Performance Engineering - 9th European Workshop, EPEW 2012, Munich, Germany, July 30, 2012, and 28th UK Workshop, UKPEW 2012, Edinburgh, UK, July 2, 2012, Revised Selected Papers*. Munich: Springer, 2012, pp. 32–47. DOI: 10.1007/978-3-642-36781-6_3 (on p. 20, 61, 75).

[4] R. A. Hayden, A. Stefanek and J. T. Bradley. “Fluid computation of passage-time distributions in large Markov models”. In: *Theoretical Computer Science* 413.1 (Jan. 2012), pp. 106–141. ISSN: 03043975. DOI: 10.1016/j.tcs.2011.07.017 (on p. 17–19, 35, 53, 58, 85, 97, 104, 105, 123, 124, 151, 169).

[5] M. Kohut, A. Stefanek, R. A. Hayden and J. T. Bradley. “Specification and efficient computation of passage-time distributions in GPA”. In: *Proceeding QEST ’12 Proceedings of the 2012 Ninth International Conference on Quantitative Evaluation of SysTems*. London, 2012, pp. 199–200. DOI: 10.1109/QEST.2012.24 (on p. 18, 20, 156).

[6] A. Stefanek, M. C. Guenther and J. T. Bradley. “Normal and inhomogeneous moment closures for stochastic process algebras”. In: *10th Workshop on Process Algebra and Stochastically Timed Activities (PASTA’11)*. Ragusa, 2011 (on p. 21, 61).

[7] A. Stefanek, U. Harder and J. T. Bradley. “Energy Consumption in the Office”. In: *UKPEW’12, 28th UK Performance Engineering Workshop, Edinburgh, UK, July 2, 2012, Revised Selected Papers*. Ed. by M. Tribastone and S. Gilmore. Vol. 7587. Lecture Notes in Computer Science. Springer, 2012, pp. 224–236. ISBN: 978-3-642-36780-9. DOI: 10.1007/978-3-642-36781-6_16 (on p. 21, 83).

[8] A. Stefanek, R. A. Hayden and J. T. Bradley. “A new tool for the performance analysis of massively parallel computer systems”. In: *Eighth Workshop on Quantitative Aspects of Programming Languages QAPL 2010 March 2728 2010 Paphos Cyprus*. Electronic Proceedings in Theoretical Computer Science (2010). DOI: 10.4204/EPTCS.28.11 (on p. 19, 61, 156).

[9] A. Stefanek, R. A. Hayden and J. T. Bradley. “Fluid Analysis of Energy Consumption using Rewards in Massively Parallel Markov Models”. In: *ICPE’11 - Second Joint WOSP/SIPEW International Conference on Performance Engineering, Karlsruhe, Germany, March 14-16, 2011*. ACM Press, 2011, p. 121. ISBN: 9781450305198. DOI: 10.1145/1958746.1958767 (on p. 20, 81).

[10] A. Stefanek, R. A. Hayden and J. T. Bradley. “Fluid computation of the performance-energy trade-off in large scale Markov models”. In: *SIGMETRICS Perform. Eval. Rev.* 39.3 (2011). DOI: 10.1145/2160803.2160872 (on p. 19, 81, 110, 129).

[11] A. Stefanek, R. A. Hayden and J. T. Bradley. “GPA - A Tool for Fluid Scalability Analysis of Massively Parallel Systems”. In: *2011 Eighth International Conference on Quantitative Evaluation of SysTems*. IEEE, Sept. 2011, pp. 147–148. ISBN: 978-1-4577-0973-9. DOI: 10.1109/QEST.2011.26 (on p. 20, 156).

[12] A. Stefanek, R. A. Hayden and J. T. Bradley. “GPA - a tool for rapid analysis of very large scale PEPA models”. In: *UKPEW’10, 26th UK Performance Engineering Workshop. 7-8th July, University of Warwick*. 2010, pp. 91–101 (on p. 21, 156).

[13] A. Stefanek, R. A. Hayden and J. T. Bradley. “Hybrid analysis of large scale PEPA models”. In: *9th Workshop on Process Algebra and Stochastically Timed Activities (PASTA)*. 2010, p. 29 (on p. 21, 61).

[14] A. Stefanek, R. A. Hayden and J. T. Bradley. “Mean-field Analysis of Large Scale Markov Fluid Models with Fluid Dependent and Time-Inhomogeneous Rates”. In: *Annals of Operations Research* to appear (2013) (on p. 19, 20, 104, 197, 198).

[15] A. Stefanek, R. A. Hayden, M. M. Gonagle and J. T. Bradley. “Mean-Field Analysis of Markov Models with Reward Feedback”. In: *Analytical and Stochastic Modeling Techniques and Applications - 19th International Conference, ASMTA 2012, Grenoble, France, June 4-6, 2012. Proceedings*. Springer, 2012, pp. 193–211. DOI: 10.1007/978-3-642-30782-9_14 (on p. 19, 20, 104).

Bibliography

[16] O. E. Akman, F. Ciocchetta, A. Degasperi and M. L. Guerriero. “Modelling Biological Clocks with Bio-PEPA: Stochasticity and Robustness for the Neurospora crassa Circadian Network”. In: *CMSB ’09 Proceedings of the 7th International Conference on Computational Methods in Systems Biology*. Ed. by P. Degano and R. Gorrieri. Vol. 5688. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, Aug. 2009, pp. 52–67. ISBN: 978-3-642-03844-0. DOI: 10.1007/978-3-642-03845-7 (on p. 32).

[17] A. Ale, P. Kirk and M. Stumpf. “A general moment expansion method for stochastic kinetic models”. In: *The Journal of chemical physics* March (2013), pp. 1–13. arXiv:arXiv:1303.5848v1 (on p. 16, 29, 30, 168).

[18] D. F. Anderson. “A modified next reaction method for simulating chemical systems with time dependent propensities and delays.” en. In: *The Journal of chemical physics* 127.21 (Dec. 2007), p. 214107. ISSN: 0021-9606. DOI: 10.1063/1.2799998 (on p. 108).

[19] H. Andersson and T. Britton. “Stochastic epidemics in dynamic populations: quasi-stationarity and extinction”. In: *Journal of Mathematical Biology* 41.6 (2000), pp. 559–580. ISSN: 0303-6812. DOI: 10.1007/s002850000060 (on p. 31).

[20] *ANTLRv3 project*. URL: http://www.antlr3.org/ (visited on 09/08/2013) (on p. 166).

[21] D. Ardagna, B. Panicucci, M. Trubian and L. Zhang. “Energy-Aware Autonomic Resource Allocation in Multitier Virtualized Environments”. In: *IEEE Transactions on Services Computing* 5.1 (Jan. 2012), pp. 2–19. ISSN: 1939-1374. DOI: 10.1109/TSC.2010.42 (on p. 36).

[22] S. Asmussen, O. Nerman and M. Olsson. “Fitting Phase-type Distributions via the EM Algorithm”. In: *Scandinavian Journal of Statistics* 23.4 (1996), pp. 419 –441 (on p. 172).

[23] S. Asmussen. *Applied Probability and Queues*. Springer New York, 2003. ISBN: 978-0-387-00211-8. DOI: 10.1007/b97236 (on p. 170).

[24] F. Baccelli, D. McDonald and J. Reynier. “A mean-field model for multiple TCP connections through a buffer implementing RED”. In: *Performance Evaluation* 49.1-4 (Sept. 2002), pp. 77–97. ISSN: 01665316. DOI: 10.1016/S0166-5316(02)00136-0 (on p. 30).

[25] F. Baccelli, A. Chaintreau, D. De Vleeschauwer and D. McDonald. “A mean-field analysis of short lived interacting TCP flows”. In: *ACM SIGMETRICS Performance Evaluation Review* 32.1 (June 2004), p. 343. ISSN: 01635999. DOI: 10.1145/1012888.1005727 (on p. 16, 30).

[26] G. Balbo. “Introduction to Stochastic Petri Nets”. In: Lecture Notes in Computer Science 2090 (Sept. 2001). Ed. by E. Brinksma, H. Hermanns and J.-P. Katoen. DOI: 10.1007/3-540-44667-2 (on p. 16, 25).

[27] L. A. Barroso and U. Hölzle. “The Case for Energy-Proportional Computing”. In: *Computer* 40.12 (Dec. 2007), pp. 33–37. ISSN: 0018-9162. DOI: 10.1109/MC.2007.443 (on p. 13, 14).

[28] F. Baskett, K. M. Chandy, R. R. Muntz and F. G. Palacios. “Open, Closed, and Mixed Networks of Queues with Different Classes of Customers”. In: *Journal of the ACM* 22.2 (Apr. 1975), pp. 248–260. ISSN: 00045411. DOI: 10.1145/321879.321887 (on p. 30).

[29] R. J. Baxter. *Exactly Solved Models in Statistical Mechanics*. Academic Press, 1982 (on p. 30).

[30] N. Bellomo and M. Pulvirenti. *Modeling in applied sciences - A kinetic theory approach*. Springer, 2000 (on p. 30).

[31] M. Benaïm and J.-Y. Le Boudec. “A class of mean field interaction models for computer and communication systems”. In: *Performance Evaluation* 65.11-12 (Nov. 2008), pp. 823–838. ISSN: 01665316. DOI: 10.1016/j.peva.2008.03.005 (on p. 16, 30, 168).

[32] P. Billingsley. *Convergence of Probability Measures*. John Wiley & Sons, 1968 (on p. 115).

[33] A. Bobbio, M. Gribaudo and M. Telek. “Analysis of Large Scale Interacting Systems by Mean Field Method”. English. In: *2008 Fifth International Conference on Quantitative Evaluation of Systems* 978 (Sept. 2008), pp. 215–224. DOI: 10.1109/QEST.2008.47 (on p. 16, 30, 168).

[34] L. Bortolussi. “Hybrid Behaviour of Markov Population Models”. In: (Nov. 2012). arXiv:1211.1643 (on p. 38).

[35] L. Bortolussi. “On the Approximation of Stochastic Concurrent Constraint Programming by Master Equation”. In: *Electronic Notes in Theoretical Computer Science* 220.3 (2008), pp. 163–180. ISSN: 15710661. DOI: 10.1016/j.entcs.2008.11.025 (on p. 32).

[36] L. Bortolussi. “Stochastic Concurrent Constraint Programming”. In: *Electronic Notes in Theoretical Computer Science* 164.3 (Oct. 2006), pp. 65–80. ISSN: 15710661. DOI: 10.1016/j.entcs.2006.07.012 (on p. 16, 25, 32).

[37] L. Bortolussi, V. Galpin and J. Hillston. “HYPE with stochastic events”. In: *Electronic Proceedings in Theoretical Computer Science* 57.Qapl (July 2011), pp. 120–133. ISSN: 2075-2180. DOI: 10.4204/EPTCS.57.9 (on p. 37).

[38] L. Bortolussi and R. A. Hayden. “Bounds on the Deviation of Discrete-time Markov Chains from Their Mean-field Model”. In: *Perform. Eval.* 70.10 (Oct. 2013), pp. 736–749. ISSN: 0166-5316. DOI: 10.1016/j.peva.2013.08.012 (on p. 61).

[39] L. Bortolussi and J. Hillston. “Fluid Approximation of CTMC with Deterministic Delays”. English. In: *2012 Ninth International Conference on Quantitative Evaluation of Systems*. IEEE, Sept. 2012, pp. 53–62. ISBN: 978-1-4673-2346-8. DOI: 10.1109/QEST.2012.13 (on p. 175).

[40] L. Bortolussi and J. Hillston. “Fluid Model Checking”. In: *CONCUR 2012 – Concurrency Theory*. Ed. by M. Koutny and I. Ulidowski. Vol. 7454. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 333–347. ISBN: 978-3-642-32939-5. DOI: 10.1007/978-3-642-32940-1_24 (on p. 35).

[41] L. Bortolussi and A. Policriti. “Dynamical Systems and Stochastic Programming: To Ordinary Differential Equations and Back”. In: *Transactions on Computational Systems Biology XI*. Lecture Notes in Computer Science 5750 (2009). Ed. by C. Priami, R.-J. Back and I. Petre, pp. 216–267. DOI: 10.1007/978-3-642-04186-0 (on p. 16, 32, 168).

[42] J.-Y. L. Boudec, D. McDonald and J. Mundinger. “A Generic Mean Field Convergence Result for Systems of Interacting Objects”. In: *Fourth International Conference on the Quantitative Evaluation of Systems (QEST 2007)*. IEEE, Sept. 2007, pp. xii–xii. ISBN: 0-7695-2883-X. DOI: 10.1109/QEST.2007.3 (on p. 16, 30, 168).

[43] J. T. Bradley, S. T. Gilmore and J. Hillston. “Analysing distributed Internet worm attacks using continuous state-space approximation of process algebra models”. In: *Journal of Computer and System Sciences* 74.6 (Sept. 2008), pp. 1013–1032. ISSN: 00220000. DOI: 10.1016/j.jcss.2007.07.005 (on p. 31).

[44] J. Bradley, N. Dingle, S. Gilmore and W. Knottenbelt. “Derivation of passage-time densities in PEPA models using ipc: the imperial PEPA compiler”. In: *11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems, 2003. MASCOTS 2003.* IEEE Comput. Soc, 2003, pp. 344–351. ISBN: 0-7695-2039-1. DOI: 10.1109/MASCOT.2003.1240679 (on p. 25).

[45] J. Bradley, N. Dingle, P. Harrison and W. Knottenbelt. “Distributed computation of passage time quantiles and transient state distributions in large semi-Markov models”. English. In: *Proceedings International Parallel and Distributed Processing Symposium*. IEEE Comput. Soc, 2003, p. 8. ISBN: 0-7695-1926-1. DOI: 10.1109/IPDPS.2003.1213505 (on p. 35, 39).

[46] J. C. Butcher. *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons, 2003. ISBN: 978-0-471-96758-3 (on p. 52).

[47] M. Cain. “The moment-generating function of the minimum of bivariate normal random variables”. In: *The American Statistician* 48.2 (1994), pp. 124–125. DOI: 10.1080/00031305.1994.10476039 (on p. 73).

[48] L. Cardelli. “On process rate semantics”. In: *Theoretical Computer Science* 391.3 (2008), pp. 190 –215. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2007.11.012 (on p. 16, 32, 33).

[49] G. Casale. “Exact analysis of performance models by the Method of Moments”. In: *Performance Evaluation* 68.6 (2011), pp. 487 –506. ISSN: 0166-5316. DOI: 10.1016/j.peva.2010.12.009 (on p. 31).

[50] F. Castella, G. Dujardin and B. Sericola. “Moments’ Analysis in Homogeneous Markov Reward Models”. In: *Methodology and Computing in Applied Probability* 11.4 (May 2008), pp. 583–601. ISSN: 1387-5841. DOI: 10.1007/s11009-008-9075-5 (on p. 34).

[51] A. Chaintreau, J.-Y. Le Boudec and N. Ristanovic. “The Age of Gossip: Spatial Mean Field Regime”. In: *ACM SIGMETRICS Performance Evaluation Review* 37.1 (June 2009), pp. 109–120. ISSN: 0163-5999. DOI: 10.1145/2492101.1555363 (on p. 105).

[52] K. M. Chandy and D. Neuse. “Linearizer: A Heuristic Algorithm for Queueing Network Models of Computing Systems”. In: *Communications of the ACM* 25.2 (Feb. 1982), pp. 126–134. ISSN: 0001-0782. DOI: 10.1145/358396.358403 (on p. 31).

[53] G. Ciardo, D. Nicol and K. Trivedi. “Discrete-event simulation of fluid stochastic Petri nets”. In: *Proceedings of the Seventh International Workshop on Petri Nets and Performance Models*. IEEE Comput. Soc, 1997, pp. 217–225. ISBN: 0-8186-7931-X. DOI: 10.1109/PNPM.1997.595553 (on p. 37).

[54] F. Ciocchetta, A. Degasperi, J. K. Heath and J. Hillston. “Modelling and analysis of the NF-κB pathway in Bio-PEPA”. In: *Transactions on Computational Systems Biology XII*. Ed. by C. Priami, R. Breitling, D. Gilbert, M. Heiner and A. Uhrmacher. Springer Berlin Heidelberg, 2010, pp. 229–262. DOI: 10.1007/978-3-642-11712-1_7 (on p. 32).

[55] F. Ciocchetta, A. Duguid, S. Gilmore, M. L. Guerriero and J. Hillston. “The Bio-PEPA Tool Suite”. In: *2009 Sixth International Conference on the Quantitative Evaluation of Systems* (Sept. 2009), pp. 309–310. DOI: 10.1109/QEST.2009.27 (on p. 32).

[56] F. Ciocchetta and J. Hillston. “Bio-PEPA: A framework for the modelling and analysis of biological systems”. In: *Theoretical Computer Science* 410.33-34 (Aug. 2009), pp. 3065–3084. ISSN: 03043975. DOI: 10.1016/j.tcs.2009.02.037 (on p. 31, 168).

[57] F. Ciocchetta and J. Hillston. “Bio-PEPA for Epidemiological Models”. In: *Electronic Notes in Theoretical Computer Science* 261 (Feb. 2010), pp. 43–69. ISSN: 15710661. DOI: 10.1016/j.entcs.2010.01.005 (on p. 32).

[58] A. Clark. “The ipclib PEPA Library”. In: *QEST’07, 4th International Conference on the Quantitative Evaluation of Systems*. IEEE, Sept. 2007, pp. 55–56. DOI: 10.1109/QEST.2007.20 (on p. 39).

[59] A. Clark, S. Gilmore and M. Tribastone. “Scalable Analysis of Scalable Systems”. In: *FASE 2009, 12th International Conference on Fundamental Approaches to Software Engineering*. Ed. by M. and Chechik and M. Wirsing. Vol. 5503. Lecture Notes in Computer Science. Springer, 2009, pp. 1–17. DOI: 10.1007/978-3-642-00593-0_1 (on p. 36).

[60] E. A. Coddington and N. Levinson. *Theory of Ordinary Differential Equations*. McGraw-Hill Book Company, 1955 (on p. 108).

[61] *Commons Math: The Apache Commons Mathematics Library*. URL: http://commons.apache.org/proper/commons-math/ (visited on 09/08/2013) (on p. 52, 162).

[62] P. Cremonesi, P. Schweitzer and G. Serazzi. “A unifying framework for the approximate solution of closed multiclass queuing networks”. In: *IEEE Transactions on Computers* 51.12 (2002), pp. 1423–1434. ISSN: 0018-9340. DOI: 10.1109/TC.2002.1146708 (on p. 31).

[63] R. W. R. Darling and J. R. Norris. “Differential equation approximations for Markov chains”. In: *Probability Surveys* 5 (Oct. 2008), pp. 37–79. ISSN: 1549-5787. DOI: 10.1214/07-PS121. arXiv:0710.3269 (on p. 58, 61).

[64] M. H. A. Davis. *Markov models and optimization*. Chapman & Hall/CRC, 1993 (on p. 107).

[65] D. D. Deavours, G. Clark, T. Courtney, D. Daly, S. Derisavi, J. M. Doyle, W. H. Sanders and P. G. Webster. “The Möbius Framework and its Implementation”. In: *IEEE Transactions on Software Engineering* 28.10 (Oct. 2002), pp. 956–969. DOI: 10.1109/TSE.2002.1041052 (on p. 25, 38).

[66] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall and W. Vogels. “Dynamo”. In: *ACM SIGOPS Operating Systems Review* 41.6 (Oct. 2007), p. 205. ISSN: 01635980. DOI: 10.1145/1323293.1294281 (on p. 13).

[67] E. de Souza e Silva and R. Gail. “An algorithm to calculate transient distributions of cumulative rate and impulse based reward”. In: *Communications in Statistics: Stochastic Models* 14.3 (1998), pp. 509–536. DOI: 10.1080/15326349808807486 (on p. 34).

[68] J. Ding. “Structural and Fluid Analysis for Large Scale PEPA Models -with Applications to Content Adaptation Systems”. PhD thesis. University of Edinburgh, 2010 (on p. 31, 34, 35, 53, 86, 168).

[69] L. Donatiello and V. Grassi. “On Evaluating the Cumulative Performance Distribution of Fault-Tolerant Computer Systems”. In: *IEEE Transactions on Computers* 40.11 (1991). ISSN: 0018-9340. DOI: 10.1109/12.102838 (on p. 34).

[70] J. Dormand and P. Prince. “A family of embedded Runge-Kutta formulae”. In: *Journal of Computational and Applied Mathematics* 6.1 (1980), pp. 19 –26. ISSN: 0377-0427. DOI: 10.1016/0771-050X(80)90013-3 (on p. 52).

[71] S. Engblom. “Computing the moments of high dimensional solutions of the master equation”. In: *Applied Mathematics and Computation* 180.2 (2006), pp. 498 –515. DOI: 10.1016/j.amc.2005.12.032 (on p. 16, 29, 30, 46, 168).

[72] S. N. Ethier and T. G. Kurtz. *Markov Processes: Characterization and Convergence*. Wiley, 2005 (on p. 31, 108, 197, 198).

[73] J. Fourneau, L. Kloul and F. Valois. “Performance modelling of hierarchical cellular networks using PEPA”. In: *Performance Evaluation* 50.2-3 (Nov. 2002), pp. 83–99. ISSN: 01665316. DOI: 10.1016/S0166-5316(02)00101-3 (on p. 25).

[74] M. Fujita, P. McGeer and J.-Y. Yang. “Multi-Terminal Binary Decision Diagrams: An Efficient Data Structure for Matrix Representation”. en. In: *Formal Methods in System Design* 10.2-3 (Apr. 1997), pp. 149–169. ISSN: 1572-8102. DOI: 10.1023/A:1008647823331 (on p. 25).

[75] V. Galpin, L. Bortolussi and J. Hillston. “HYPE: hybrid modelling by composition of flows”. In: *Formal Aspects of Computing* 25.4 (2010), pp. 1–39. DOI: 10.1007/s00165-011-0189-0 (on p. 37).

[76] A. Gandhi, V. Gupta and M. Harchol-Balter. “Optimality analysis of energy-performance trade-off for server farm management”. In: *Performance Evaluation* 67.11 (2010), pp. 1155–1171. DOI: 10.1016/j.peva.2010.08.009 (on p. 36).

[77] A. Gandhi and M. Harchol-Balter. “The case for sleep states in servers”. In: *HotPower ’11 Proceedings of the 4th Workshop on Power-Aware Computing and Systems*. 2011. ISBN: 9781450309813. DOI: 10.1145/2039252.2039254 (on p. 36).

[78] A. Gandhi, Yuan Chen, D. Gmach, M. Arlitt and M. Marwah. “Minimizing data center SLA violations and power consumption via hybrid resource provisioning”. English. In: *2011 International Green Computing Conference and Workshops*. IEEE, July 2011, pp. 1–8. ISBN: 978-1-4577-1222-7. DOI: 10.1109/IGCC.2011.6008611 (on p. 13, 36).

[79] N. Gast and B. Gaujal. *A mean field model of work stealing in large-scale systems*. Vol. 38. 1. New York, New York, USA: ACM Press, June 2010, p. 13. ISBN: 9781450300384. DOI: 10.1145/1811039.1811042 (on p. 16).

[80] E. Gelenbe and C. Morfopoulou. "Power Savings in Packet Networks via Optimised Routing". In: *Mobile Networks and Applications* 17.1 (Oct. 2011), pp. 152–159. ISSN: 1383-469X. DOI: 10.1007/s11036-011-0344-0 (on p. 36).

[81] C. S. Gillespie. "Moment-closure approximations for mass-action models". In: *IET Systems Biology* 3.1 (2009), pp. 52–58 (on p. 16, 29, 30, 39, 110, 168).

[82] D. T. Gillespie. "Exact stochastic simulation of coupled chemical reactions". In: *Journal of Physical Chemistry* 81.25 (1977), pp. 2340–2361. ISSN: 00223654. DOI: 10.1021/j100540a008 (on p. 24, 39, 45).

[83] D. T. Gillespie. *Markov Processes: An Introduction for Physical Scientist*. Gulf Professional Publishing, 1992, p. 565. ISBN: 0122839552 (on p. 29, 34).

[84] S. Gilmore, J. Hillston and M. Ribaudo. "An efficient algorithm for aggregating PEPA models". In: *IEEE Transactions on Software Engineering* 27.5 (May 2001), pp. 449–464. ISSN: 00985589. DOI: 10.1109/32.922715 (on p. 28).

[85] S. Gilmore, J. Hillston, R. Holton and M. Rettelbach. "Specifications in Stochastic Process Algebra for a Robot Control Problem". In: *International Journal of Production Research* (1995) (on p. 25).

[86] R. Gonzalez and M. Horowitz. "Energy dissipation in general purpose microprocessors". In: *IEEE Journal of Solid-State Circuits* 31.9 (1996), pp. 1277–1284. ISSN: 00189200. DOI: 10.1109/4.535411 (on p. 36).

[87] *GPAnalyser project*. URL: http://code.google.com/p/gpanalyser/ (visited on 09/08/2013) (on p. 166).

[88] C. Graham. "Kinetic limits for large communication networks". In: *Modeling in applied sciences: a kinetic theory approach* (2000), pp. 317–330 (on p. 16, 30).

[89] C. Graham and P. Robert. "Interacting multi-class transmissions in large stochastic networks". EN. In: *The Annals of Applied Probability* 19.6 (Dec. 2009), pp. 2334–2361. ISSN: 2168-8737. DOI: 10.1214/09-AAP614 (on p. 16, 30).

[90] M. Gribaudo and M. Telek. "Fluid models in performance analysis". In: *Formal Methods for Performance Evaluation*. Ed. by M. Bernardo and J. Hillston. Vol. 4486. Springer Berlin Heidelberg, May 2007, pp. 271–317. ISBN: 978-3-540-72482-7. DOI: 10.1007/978-3-540-72522-0_7 (on p. 37).

[91] M. Gribaudo, M. Sereno and A. Bobbio. "Fluid Stochastic Petri Nets: An Extended Formalism to Include Non-Markovian Models". In: *Proceedings of The 8th International Workshop on Petri Nets and Performance Models*. Zaragoza: IEEE Computer Society, Sept. 1999, pp. 74 –81. ISBN: 0-7695-0331-4. DOI: 10.1109/PNPM.1999.796554 (on p. 37).

[92] M. C. Guenther and J. T. Bradley. “Higher moment analysis of a spatial stochastic process algebra.” In: *Proceedings of the 8th European conference on Computer Performance Engineering*. 2011, pp. 87–101. DOI: 10.1007/978-3-642-24749-1_8 (on p. 156, 160, 169).

[93] P. G. Harrison and W. J. Knottenbelt. “Passage Time Distributions in Large Markov Chains”. In: *Proceedings of the ACM SIGMETRICS international conference on Measurement and modeling of computer systems - SIGMETRICS ’02*. Vol. 30. ACM SIGMETRICS Performance Evaluation Review 1. May 2002, pp. 77–85. DOI: 10.1145/511399.511345 (on p. 35).

[94] J. Hasenauer, V. Wolf, A. Kazeroonian and F. J. Theis. “Method of conditional moments (MCM) for the Chemical Master Equation : A unified framework for the method of moments and hybrid stochastic-deterministic models.” In: *Journal of mathematical biology* (Aug. 2013). ISSN: 1432-1416. DOI: 10.1007/s00285-013-0711-5 (on p. 38).

[95] R. A. Hayden. “Mean-field approximations for performance models with generally-timed transitions”. In: *SIGMETRICS Perform. Eval. Rev.* 39.3 (2011), pp. 119–121. DOI: 10.1145/2160803.2160877 (on p. 46, 175).

[96] R. A. Hayden. “Mean Field for Performance Models with Deterministically-Timed Transitions”. In: *2012 Ninth International Conference on Quantitative Evaluation of Systems*. London: IEEE, Sept. 2012, pp. 63–73. ISBN: 978-1-4673-2346-8. DOI: 10.1109/QEST.2012.27 (on p. 175).

[97] R. A. Hayden. “Mean-field models for interacting battery-powered devices”. In: *Imperial College Energy and Performance Colloqium*. 2012 (on p. 105, 175).

[98] R. A. Hayden. “Scalable performance analysis of massively parallel stochastic systems”. PhD thesis. 2011 (on p. 62).

[99] R. A. Hayden and J. T. Bradley. “A fluid analysis framework for a Markovian process algebra”. In: *Theoretical Computer Science* 411.22-24 (May 2010), pp. 2260–2297. ISSN: 03043975. DOI: 10.1016/j.tcs.2010.02.001 (on p. 16, 17, 29–32, 40, 48, 63, 109, 110, 156, 168).

[100] R. A. Hayden and J. T. Bradley. “A functional central limit theorem for PEPA”. In: *PASTA’09, Proceedings of the 8th Workshop on Process Algebra and Stochastically Timed Activities*. Edinburgh, Aug. 2009, pp. 13–23 (on p. 31, 58).

[101] R. A. Hayden and J. T. Bradley. “Evaluating fluid semantics for passive stochastic process algebra cooperation”. In: *Performance Evaluation* 67.4 (2010), pp. 260–284. DOI: 10.1016/j.peva.2009.08.010 (on p. 31).

[102] R. A. Hayden and J. T. Bradley. *Shared Transaction Markov Chains for fluid analysis of massively parallel systems*. IEEE, Sept. 2009, pp. 1–12. ISBN: 978-1-4244-4927-9. DOI: 10.1109/MASCOT.2009.5367050 (on p. 130).

[103] R. A. Hayden, J. T. Bradley and A. Clark. “Performance Specification and Evaluation with Unified Stochastic Probes and Fluid Analysis”. In: *IEEE Transactions on Software Engineering* 39.1 (Jan. 2013), pp. 97–118. ISSN: 0098-5589. DOI: 10.1109/TSE.2012.1 (on p. 21, 34, 35, 55, 82, 123, 164).

[104] T. A. Henzinger. “The theory of hybrid automata”. In: *Logic in Computer Science, 1996. LICS ’96. Proceedings., Eleventh Annual IEEE Symposium on*. IEEE Computer Society, July 1996, pp. 278–292. ISBN: 0-8186-7463-6. DOI: 10.1109/LICS.1996.561342 (on p. 37).

[105] S. Herbert and D. Marculescu. “Analysis of dynamic voltage/frequency scaling in chip-multiprocessors”. In: *Proceedings of the 2007 international symposium on Low power electronics and design - ISLPED ’07*. New York, New York, USA: ACM Press, 2007, pp. 38–43. ISBN: 9781595937094. DOI: 10.1145/1283780.1283790 (on p. 14).

[106] H. Hermanns, M. Kwiatkowska, G. Norman, D. Parker and M. Siegle. “On the use of MTBDDs for performability analysis and verification of stochastic systems”. In: *The Journal of Logic and Algebraic Programming* 56.1-2 (May 2003), pp. 23–67. ISSN: 15678326. DOI: 10.1016/S1567-8326(02)00066-8 (on p. 25).

[107] J. Hillston. *A compositional approach to performance modelling*. Cambridge University Press, Oct. 1996. ISBN: 0-521-57189-8 (on p. 16, 25).

[108] J. Hillston. “Fluid flow approximation of PEPA models”. In: *QEST*. IEEE, Sept. 2005, pp. 33–42. ISBN: 0-7695-2427-3. DOI: 10.1109/QEST.2005.12 (on p. 16, 31, 39, 109, 168).

[109] G. Horton, V. G. Kulkarni, D. M. Nicol and K. S. Trivedi. “Fluid stochastic Petri nets: Theory, applications, and solution techniques”. In: *European Journal of Operational Research* 105.1 (1998), pp. 184–201. DOI: 10.1016/S0377-2217(97)00028-3 (on p. 37).

[110] G. Horváth, S. Rácz, Á. Tari and M. Telek. “Evaluation of Reward Analysis Methods with MRMSolve 2.0.” In: *Proceedings of the First International Conference on Quantitative Evaluation of Systems*. 2004, pp. 165–174. DOI: 10.1109/QEST.2004.1348031 (on p. 34, 35).

[111] W. Huang, M. Allen-Ware, J. B. Carter, E. Elnozahy, H. Hamann, T. Keller, C. Lefurgy, K. Rajamani and J. Rubio. “TAPO: Thermal-aware power optimization techniques for servers and data centers”. English. In: *2011 International Green Computing Conference and Workshops*. IEEE, July 2011, pp. 1–8. ISBN: 978-1-4577-1222-7. DOI: 10.1109/IGCC.2011.6008610 (on p. 15).

[112] *Intel product information*. URL: http://ark.intel.com (visited on 09/09/2013) (on p. 101).

[113] L. Isserlis. “On a Formula for the Product-Moment Coefficient of any Order of a Normal Frequency Distribution in any Number of Variables”. In: *Biometrika* 12.1/2 (1918), pp. 134–139. ISSN: 00063444 (on p. 50).

[114] J. R. Jackson. “Networks of Waiting Lines”. In: *Operations Research* 5.4 (Aug. 1957), pp. 518–521. ISSN: 0030-364X. DOI: 10.1287/opre.5.4.518 (on p. 30).

[115] *Joulemeter*. URL: http://research.microsoft.com/en-us/projects/joulemeter (visited on 09/08/2013) (on p. 102).

[116] O. Kallenberg. *Foundations of Modern Probability*. Springer, 2002 (on p. 59, 197).

[117] J. G. Kemény and J. L. Snell. *Finite markov chains*. Springer-Verlag, 1976 (on p. 28).

[118] G. Kesidis, T. Konstantopoulos and P. Sousi. “A stochastic epidemiological model and a deterministic limit for BitTorrent-like peer-to-peer file-sharing networks”. In: (Nov. 2008), p. 25. arXiv:0811.1003 (on p. 42).

[119] U. Khadim. *A comparative study of process algebras for hybrid systems*. Computer Science Report 06–23. Technische Universiteit Eindhoven, 2006 (on p. 37).

[120] F. C. Klebaner. *Introduction to stochastic calculus with applications*. Second edition. Imperial College Press, 2006 (on p. 114).

[121] W. J. Knottenbelt and P. G. Harrison. “Distributed disk-based solution techniques for large Markov models”. In: *NSMC’99, Proceedings of the 3rd Intl. Conference on the Numerical Solution of Markov Chains* (1999). Ed. by B. Plateau, W. Stewart and M. Silva, pp. 58–75 (on p. 25).

[122] H. Kobayashi. “Application of the Diffusion Approximation to Queueing Networks I: Equilibrium Queue Distributions”. In: *Journal of the ACM* 21.2 (Apr. 1974), pp. 316–328. ISSN: 0004-5411. DOI: 10.1145/321812.321827 (on p. 31).

[123] H. Kobayashi. “Application of the Diffusion Approximation to Queueing Networks II: Nonequilibrium Distributions and Applications to Computer Modeling”. In: *Journal of the ACM* 21.3 (July 1974), pp. 459–469. ISSN: 0004-5411. DOI: 10.1145/321832.321844 (on p. 31).

[124] M. Kohut. *A unified performance query formalism*. Tech. rep. 2012 (on p. 164).

[125] I. Krishnarajah, A. Cook, G. Marion and G. Gibson. “Novel moment closure approximations in stochastic epidemics.” In: *Bulletin of Mathematical Biology* 67.4 (2005), pp. 855–873. DOI: 10.1016/j.bulm.2004.11.002 (on p. 30).

[126] T. G. Kurtz. “Limit theorems and diffusion approximations for density dependent Markov chains”. In: *Stochastic Systems: Modeling, Identification and Optimization, I*. Ed. by R. J.-B. Wets. Vol. 5. Mathematical Programming Studies. Springer Berlin Heidelberg, 1976, pp. 67–78. ISBN: 978-3-642-00783-5. DOI: 10.1007/BFb0120765 (on p. 31).

[127] T. G. Kurtz. “Solutions of Ordinary Differential Equations as Limits of Pure Jump Markov Processes”. In: 7.1 (Apr. 1970), pp. 49–58 (on p. 16, 30).

[128] T. G. Kurtz. “Strong approximation theorems for density dependent Markov chains”. In: *Stochastic Processes and their Applications* 6.3 (Feb. 1978), pp. 223–240. ISSN: 03044149. DOI: 10.1016/0304-4149(78)90020-0 (on p. 31, 58).

[129] M. Z. Kwiatkowska, G. Norman and D. Parker. “PRISM: Probabilistic Symbolic Model Checker”. In: (Apr. 2002), pp. 200–204 (on p. 25, 38).

[130] M. Kwiatkowski and I. Stark. “The Continuous π-Calculus: A Process Algebra for Biochemical Modelling”. In: *Computational Methods in Systems Biology*. Ed. by M. Heiner and A. Uhrmacher. Springer Berlin Heidelberg, 2008, pp. 103–122. DOI: 10.1007/978-3-540-88562-7_11 (on p. 32).

[131] M. Lapin, L. Mikeev and V. Wolf. “SHAVE: stochastic hybrid analysis of markov population models”. In: *Proceedings of the 14th international conference on Hybrid systems: computation and control*. HSCC ’11. New York, NY, USA: ACM, 2011, pp. 311–312. ISBN: 978-1-4503-0629-4. DOI: 10.1145/1967701.1967746 (on p. 39).

[132] C. H. Lee, K.-H. Kim and P. Kim. “A moment closure method for stochastic reaction networks.” In: *The Journal of chemical physics* 130.13 (2009), p. 134107. DOI: 10.1063/1.3103264. (on p. 29, 168).

[133] H. Lim, A. Kansal and J. Liu. “Power Budgeting for Virtualized Data Centers”. In: *Proceedings of the 2011 USENIX Conference on USENIX Annual Technical Conference*. USENIXATC’11. Portland, OR: USENIX Association, 2011, pp. 5–5 (on p. 102).

[134] M. Litzkow, M. Livny and M. Mutka. “Condor-a hunter of idle workstations”. In: *8th International Conference on Distributed Computing Systems* (1988), pp. 104 –111. DOI: 10.1109/DCS.1988.12507 (on p. 169).

[135] Z. Liu, Y. Chen, C. Bash, A. Wierman, D. Gmach, Z. Wang, M. Marwah and C. Hyser. “Renewable and cooling aware workload management for sustainable data centers”. In: *ACM SIGMETRICS Performance Evaluation Review* 40.1 (June 2012), p. 175. ISSN: 01635999. DOI: 10.1145/2318857.2254779 (on p. 104).

[136] R. Marler and J. Arora. “Survey of multi-objective optimization methods for engineering”. In: *Structural and Multidisciplinary Optimization* 26.6 (Apr. 2004), pp. 369–395. ISSN: 1615-147X. DOI: 10.1007/s00158-003-0368-6 (on p. 36).

[137] T. J. Martin Arlitt. “Workload Characterization of the 1998 World Cup Web Site”. In: *IEEE Network* 14.3 (2000), pp. 30–37. DOI: 10.1109/65.844498 (on p. 104, 124, 125).

[138] M. Massink, M. Brambilla, D. Latella, M. Dorigo and M. Birattari. “Analysing robot swarm decision-making with Bio-PEPA”. In: *ANTS’12 Proceedings of the 8th international conference on Swarm Intelligence*. Ed. by M. Dorigo, M. Birattari, C. Blum, A. L. Christensen, A. P. Engelbrecht, R. Groß and T. Stützle. Vol. 7461. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, Sept. 2012, pp. 25–36. ISBN: 978-3-642-32649-3. DOI: 10.1007/978-3-642-32650-9 (on p. 32).

[139] M. Massink, M. Brambilla, D. Latella, M. Dorigo and M. Birattari. “On the use of Bio-PEPA for modelling and analysing collective behaviours in swarm robotics”. In: *Swarm Intelligence* 7.2-3 (Apr. 2013), pp. 201–228. ISSN: 1935-3812. DOI: 10.1007/s11721-013-0079-6 (on p. 32).

[140] M. Massink, D. Latella, A. Bracciali and J. Hillston. “Modelling non-linear crowd dynamics in Bio-PEPA”. In: *Proceedings of the 14th International Conference on Fundamental Approaches to Software Engineering*. Ed. by D. Giannakopoulou and F. Orejas. 2011, pp. 96–110. DOI: 10.1007/978-3-642-19811-3_8 (on p. 32).

[141] M. Massink, D. Latella, A. Bracciali, M. D. Harrison and J. Hillston. “Scalable context-dependent analysis of emergency egress models”. In: *Formal Aspects of Computing* 24.2 (July 2011), pp. 267–302. ISSN: 0934-5043. DOI: 10.1007/s00165-011-0188-1 (on p. 32).

[142] T. I. Matis and I. G. Guardiola. “Achieving Moment Closure through Cumulant Neglect”. In: *Mathematica* (2010) (on p. 29).

[143] J. C. McCullough, Y. Agarwal, J. Chandrashekar, S. Kuppuswamy, A. C. Snoeren and R. K. Gupta. “Evaluating the Effectiveness of Model-based Power Characterization”. In: *Proceedings of the 2011 USENIX Conference on USENIX Annual Technical Conference*. USENIXATC’11. Portland, OR: USENIX Association, 2011, pp. 12–12 (on p. 102).

[144] B. Melamed and M. Yadin. “Randomization Procedures in the Computation of Cumulative-Time Distributions over Discrete State Markov Processes”. In: *Operations Research* 32.4 (1984), pp. 926–944. DOI: 10.1287/opre.32.4.926 (on p. 35).

[145] S. Méléard. “Asymptotic behaviour of some interacting particle systems; McKean-Vlasov and Boltzmann models”. In: *Probabilistic Models for Nonlinear Partial Differential Equations*. Lecture Notes in Mathematics 1627 (1996). Ed. by D. Talay and L. Tubaro, pp. 42–95. DOI: 10.1007/BFb0093175 (on p. 30).

[146] P. Milner, C. S. Gillespie and D. J. Wilkinson. “Moment closure approximations for stochastic kinetic models with rational rate laws.” In: *Mathematical biosciences* 231.2 (June 2011), pp. 99–104. ISSN: 1879-3134. DOI: 10.1016/j.mbs.2011.02.006 (on p. 30).

[147] C. Mobius, W. Dargie and A. Schill. “Power Consumption Estimation Models for Processors, Virtual Machines, and Servers”. In: *IEEE Transactions on Parallel and Distributed Systems* 99.PrePrints (2013), p. 1. ISSN: 1045-9219. DOI: 10.1109/TPDS.2013.183 (on p. 101, 102).

[148] J. K. Muppala and K. S. Trivedi. “Numerical transient solution of finite Markovian queueing systems”. In: *Queueing and related models*. Oxford university press, 1992, pp. 262–262 (on p. 35).

[149] H. Nabli and B. Sericola. “Performability analysis: a new algorithm”. In: *IEEE Transactions on Computers* 45.4 (Apr. 1996), pp. 491–494. ISSN: 00189340. DOI: 10.1109/12.494108 (on p. 34).

[150] J. R. Norris. *Markov chains*. Cambridge University Press, 1998, p. 237. ISBN: 0521633966 (on p. 23, 86).

[151] A. Phillips and L. Cardelli. “A Correct Abstract Machine for the Stochastic Pi-calculus”. In: *BIOCONCUR’04, Concurrent Models in Molecular Biology*. Aug. 2004 (on p. 38).

[152] A. Phillips and L. Cardelli. “Efficient, correct simulation of biological processes in stochastic Pi-calculus”. In: *CMSB’07, Proceedings of Computational Methods in Systems Biology*. Vol. 4695. Lecture Notes in Computer Science. Springer-Verlag, Sept. 2007, pp. 184–199. DOI: 10.1007/978-3-540-75140-3_13 (on p. 32, 38).

[153] B. Plateau and K. Atif. “Stochastic automata network of modeling parallel systems”. English. In: *IEEE Transactions on Software Engineering* 17.10 (1991), pp. 1093–1108. ISSN: 00985589. DOI: 10.1109/32.99196 (on p. 25).

[154] M. J. D. Powell. “Direct search algorithms for optimization calculations”. English. In: *Acta Numerica* 7 (Nov. 2008), p. 287. ISSN: 0962-4929. DOI: 10.1017/S0962492900002841 (on p. 101).

[155] C. Priami. “Stochastic pi-Calculus”. In: *The Computer Journal* 38.7 (July 1995), pp. 578–589. ISSN: 0010-4620. DOI: 10.1093/comjnl/38.7.578 (on p. 16, 25).

[156] S. Ramsey, D. Orrell and H. Bolouri. “Dizzy: Stochastic Simulation of Large-scale Genetic Regulatory Networks”. In: *Journal of Bioinformatics and Computational Biology* 3.2 (2005), pp. 415–436 (on p. 39).

[157] A. Rawson, J. Pfleuger and T. Cader. “Data Center Power Efficiency Metrics: PUE and DCiE”. In: *The Green Grid* (2007) (on p. 120).

[158] M. Reiser and S. S. Lavenberg. “Mean-Value Analysis of Closed Multichain Queuing Networks”. In: *Journal of the ACM* 27.2 (Apr. 1980), pp. 313–322. ISSN: 00045411. DOI: 10.1145/322186.322195 (on p. 31).

[159] *Renewable energy – Data Centers – Google*. URL: http://www.google.co.uk/about/datacenters/renewable/index.html (visited on 14/09/2013) (on p. 14).

[160] *Report to Congress on Server and Data Center Energy Efficiency Public Law 109-431*. Tech. rep. US Environmental Protection Agency, 2007 (on p. 14).

[161] L. M. Rios and N. V. Sahinidis. “Derivative-free optimization: a review of algorithms and comparison of software implementations”. English. In: *Journal of Global Optimization* 56.3 (2013), pp. 1247–1293. ISSN: 0925-5001. DOI: 10.1007/s10898-012-9951-y (on p. 101).

[162] A. Riska and E. Smirni. “Autonomic exploration of trade-offs between power and performance in disk drives”. In: *Proceeding of the 7th international conference on Autonomic computing - ICAC ’10*. New York, New York, USA: ACM Press, June 2010, p. 131. ISBN: 9781450300742. DOI: 10.1145/1809049.1809072 (on p. 36).

[163] S. M. Ross. *Stochastic processes*. Wiley, 1996, p. 510. ISBN: 0471120626 (on p. 23, 24).

[164] P. J. Schweitzer. “Approximate analysis of multiclass closed networks of queues”. In: *Proceedings of the International Conference on Stochastic Control and Optimization*. Amsterdam, Netherlands, 1979, pp. 25–29 (on p. 31).

[165] A. B. Singer and P. I. Barton. “Global Optimization with Nonlinear Ordinary Differential Equations”. In: *Journal of Global Optimization* 34.2 (Feb. 2006), pp. 159–190. ISSN: 0925-5001. DOI: 10.1007/s10898-005-7074-4 (on p. 101).

[166] A. Singh and J. Hespanha. “Lognormal moment closures for biochemical reactions”. In: *Decision and Control 2006 45th IEEE Conference on*. 2. IEEE, 2006, pp. 2063–2068. ISBN: 1424401712. DOI: 10.1109/CDC.2006.376994 (on p. 30).

[167] J. Slegers, I. Mitrani and N. Thomas. “Evaluating the optimal server allocation policy for clusters with on/off sources”. In: *Performance Evaluation* 66.8 (Aug. 2009), pp. 453–467. ISSN: 01665316. DOI: 10.1016/j.peva.2009.01.004 (on p. 14, 36).

[168] V. Sotiropoulos and Y. N. Kaznessis. “Analytical Derivation of Moment Equations in Stochastic Chemical Kinetics.” In: *Chemical engineering science* 66.3 (Feb. 2011), pp. 268–277. ISSN: 0009-2509. DOI: 10.1016/j.ces.2010.10.024 (on p. 29).

[169] A. Stefanek. “Continuous and spatial extension of stochastic Pi-calculus”. MA thesis. Department of Computing, Imperial College London, July 2009 (on p. 38).

[170] C. Stewart, D. Gmach and M. Arlitt. “Policy and mechanism for carbon-aware cloud applications”. In: *2012 IEEE Network Operations and Management Symposium*. IEEE, Apr. 2012, pp. 590–594. ISBN: 978-1-4673-0269-2. DOI: 10.1109/NOMS.2012.6211963 (on p. 14).

[171] D. J. Sumpter, G. B. Blanchard and D. S. Broomhead. “Ants and agents: a process algebra approach to modelling ant colony behaviour.” In: *Bulletin of mathematical biology* 63.5 (Sept. 2001), pp. 951–80. ISSN: 0092-8240. DOI: 10.1006/bulm.2001.0252 (on p. 16).

[172] Q. Tang, S. Gupta and G. Varsamopoulos. “Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach”. In: *IEEE Transactions on Parallel and Distributed Systems* 19.11 (2008), pp. 1458–1472. DOI: 10.1109/TPDS.2008.111 (on p. 104).

[173] Á. Tari, M. Telek and P. Buchholz. “A Unified Approach to the Moments Based Distribution Estimation - Unbounded Support”. In: *Formal Techniques for Computer Systems and Business Processes*. Ed. by M. Bravetti, L. Kloul and G. Zavattaro. Vol. 3670. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 79–93. ISBN: 978-3-540-28701-8. DOI: 10.1007/11549970 (on p. 82, 93, 95, 162).

[174] M. Telek, A. Horváth and G. Horváth. “Analysis of inhomogeneous Markov reward models”. In: *Linear Algebra and its Applications* 386 (July 2004), pp. 383–405. ISSN: 00243795. DOI: 10.1016/j.laa.2004.02.002 (on p. 34, 35).

[175] M. Telek and S. Rácz. “Numerical Analysis of Large Markov Reward Models”. In: *Performance Evaluation* 36-37.1-4 (1999), pp. 95–114. DOI: 10.1016/S0166-5316(99)00032-2 (on p. 34, 81).

[176] D. Thain, T. Tannenbaum and M. Livny. “Distributed computing in practice: the Condor experience.” In: *Concurrency - Practice and Experience* 17.2-4 (2005), pp. 323–356. DOI: 10.1002/cpe.v17:2/4 (on p. 167).

[177] *The MIT License*. URL: http://opensource.org/licenses/MIT (visited on 14/08/2013) (on p. 166).

[178] N. Thomas and Y. Zhao. “Mean value analysis for a class of PEPA models”. In: *The Computer Journal* 54.5 (Sept. 2010), pp. 643–652. ISSN: 0010-4620. DOI: 10.1093/comjnl/bxq064 (on p. 31).

[179] H. Tijms and R. Veldman. “A fast algorithm for the transient reward distribution in continuous-time Markov chains”. In: *Operations Research Letters* 26.4 (May 2000), pp. 155–158. ISSN: 01676377. DOI: 10.1016/S0167-6377(00)00023-7 (on p. 34).

[180] M. Tribastone. “A Fluid Model for Layered Queueing Networks”. English. In: *IEEE Transactions on Software Engineering* 39.6 (June 2013), pp. 744–756. ISSN: 0098-5589. DOI: 10.1109/TSE.2012.66 (on p. 33).

[181] M. Tribastone, J. Ding, S. Gilmore and J. Hillston. “Fluid Rewards for a Stochastic Process Algebra”. In: *IEEE Transactions on Software Engineering* 38.4 (July 2012), pp. 861–874. ISSN: 0098-5589. DOI: 10.1109/TSE.2011.81 (on p. 34, 35, 53).

[182] M. Tribastone, A. Duguid and S. Gilmore. “The PEPA eclipse plugin”. In: *ACM SIGMETRICS Performance Evaluation Review* 36.4 (Mar. 2009), p. 28. ISSN: 01635999. DOI: 10.1145/1530873.1530880 (on p. 39).

[183] M. Tribastone, S. T. Gilmore and J. Hillston. “Scalable Differential Analysis of Process Algebra Models”. In: *IEEE Transactions on Software Engineering* 38.1 (Jan. 2012), pp. 205–219. ISSN: 0098-5589. DOI: 10.1109/TSE.2010.82 (on p. 16, 31, 109, 168).

[184] K. S. Trivedi and V. G. Kulkarni. “FSPNs: Fluid stochastic Petri nets”. In: *Application and Theory of Petri Nets*. Vol. 691. Springer Berlin / Heidelberg, 1993, pp. 24–31. DOI: 10.1007/3-540-56863-8_38 (on p. 37).

[185] B. Tuffin, D. S. Chen and K. S. Trivedi. “Comparison of Hybrid Systems and Fluid Stochastic PetriNets”. In: *Discrete Event Dynamic Systems* 11.1-2 (Jan. 2001), pp. 77–95. ISSN: 0924-6703. DOI: 10.1023/A:1008387132533 (on p. 37).

[186] N. G. Van Kampen. *Stochastic Processes in Physics and Chemistry*. Ed. by S. Albeverio, P. Combe and M. Sirugue-Collin. Vol. 11. North-Holland personal library 2. North-Holland, 1992, p. 465. ISBN: 0444893490. DOI: 10.2307/2984076 (on p. 16, 29, 168).

[187] H. Wang, K. C. Sevcik, G. Serazzi and S. Wang. “The general form linearizer algorithms: A new family of approximate mean value analysis algorithms”. In: *Performance Evaluation* 65.2 (2008), pp. 129 –151. ISSN: 0166-5316. DOI: 10.1016/j.peva.2007.05.005 (on p. 31).

[188] J. Wang and C. Liu. “Generating multivariate mixture of normal distributions using a modified Cholesky decomposition”. In: *Winter Simulation Conference* (2006), p. 342. DOI: 10.1109/WSC.2006.323100 (on p. 70).

[189] P. Whittle. “On the use of the normal approximation in the treatment of stochastic processes”. In: *Journal of the Royal Statistical Society Series B Methodological* 19.2 (1957), pp. 268–281. ISSN: 00359246 (on p. 16, 29).

[190] W. Whitt. *Internet supplement to Stochastic-Process Limits*. 2002. URL: http://www.columbia.edu/~ww2040/supplement.html (visited on 09/09/2013) (on p. 198).

[191] A. Wierman, L. L. H. Andrew and A. Tang. “Power-Aware Speed Scaling in Processor Sharing Systems”. In: *IEEE INFOCOM 2009 - The 28th Conference on Computer Communications*. IEEE, Apr. 2009, pp. 2007–2015. ISBN: 978-1-4244-3512-8. DOI: 10.1109/INFCOM.2009.5062123 (on p. 36).

[192] Y. Zhao. “Practical Applications of Performance Modelling of Security Protocols Using PEPA”. PhD thesis. Newcastle University, 2011 (on p. 62).

Appendix A

Chapter 3

A.1 ODE systems

A.1.1 First-order moments

$$
\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[C(t)] &= -\min(\widetilde{\mathbb{E}}[C(t)], \widetilde{\mathbb{E}}[S(t)])r_r + \widetilde{\mathbb{E}}[C_t(t)]r_t \\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[C_w(t)] &= \min(\widetilde{\mathbb{E}}[C(t)], \widetilde{\mathbb{E}}[S(t)])r_r - \min(\widetilde{\mathbb{E}}[C_w(t)], \widetilde{\mathbb{E}}[S_g(t)])r_d \\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[C_t(t)] &= \min(\widetilde{\mathbb{E}}[C_w(t)], \widetilde{\mathbb{E}}[S_g(t)])r_d - r_t\widetilde{\mathbb{E}}[C_t(t)] \\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[S(t)] &= -\min(\widetilde{\mathbb{E}}[C(t)], \widetilde{\mathbb{E}}[S(t)])r_r + \min(\widetilde{\mathbb{E}}[C_w(t)], \widetilde{\mathbb{E}}[S_g(t)])r_d + \widetilde{\mathbb{E}}[S_b(t)]r_s - \widetilde{\mathbb{E}}[S(t)]r_b \\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[S_g(t)] &= \min(\widetilde{\mathbb{E}}[C(t)], \widetilde{\mathbb{E}}[S(t)])r_r - \min(\widetilde{\mathbb{E}}[C_w(t)], \widetilde{\mathbb{E}}[S_g(t)])r_d \\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[S_b(t)] &= -\widetilde{\mathbb{E}}[S_b(t)]r_s + \widetilde{\mathbb{E}}[S(t)]r_b
\end{aligned}
$$

A.1.2 Second-order moments

$$
\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[C(t)^2] &= -2\min(\widetilde{\mathbb{E}}[C(t)^2], \widetilde{\mathbb{E}}[C(t)S(t)])r_r + \min(\widetilde{\mathbb{E}}[C(t)], \widetilde{\mathbb{E}}[S(t)])r_r + 2\widetilde{\mathbb{E}}[C(t)C_t(t)]r_t + \widetilde{\mathbb{E}}[C_t(t)]r_t \\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[C(t)C_w(t)] &= -\min(\widetilde{\mathbb{E}}[C(t)C_w(t)], \widetilde{\mathbb{E}}[S(t)C_w(t)])r_r + \min(\widetilde{\mathbb{E}}[C(t)^2], \widetilde{\mathbb{E}}[C(t)S(t)])r_r - \min(\widetilde{\mathbb{E}}[C(t)], \widetilde{\mathbb{E}}[S(t)])r_r \\
&\quad + \widetilde{\mathbb{E}}[C_t(t)C_w(t)]r_t - \min(\widetilde{\mathbb{E}}[C(t)C_w(t)], \widetilde{\mathbb{E}}[C(t)S_g(t)])r_d \\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[C(t)C_t(t)] &= -\min(\widetilde{\mathbb{E}}[C(t)C_t(t)], \widetilde{\mathbb{E}}[S(t)C_t(t)])r_r - \widetilde{\mathbb{E}}[C(t)C_t(t)]r_t \\
&\quad + \widetilde{\mathbb{E}}[C_t(t)C_w(t)]r_t + \widetilde{\mathbb{E}}[C_t(t)^2]r_t - \widetilde{\mathbb{E}}[C_t(t)]r_t + \min(\widetilde{\mathbb{E}}[C_w(t)C(t)], \widetilde{\mathbb{E}}[S_g(t)C(t)])r_d \\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[C(t)S(t)] &= -\min(\widetilde{\mathbb{E}}[C(t)S(t)], \widetilde{\mathbb{E}}[S(t)^2])r_r - \min(\widetilde{\mathbb{E}}[C(t)^2], \widetilde{\mathbb{E}}[S(t)C(t)])r_r + \min(\widetilde{\mathbb{E}}[C(t)], \widetilde{\mathbb{E}}[S(t)])r_r \\
&\quad + \widetilde{\mathbb{E}}[C_t(t)C_w(t)]r_t + \widetilde{\mathbb{E}}[C_t(t)S(t)]r_t + \widetilde{\mathbb{E}}[S_b(t)C(t)]r_s - \widetilde{\mathbb{E}}[S(t)C(t)]r_b \\
&\quad + \min(\widetilde{\mathbb{E}}[C_w(t)C(t)], \widetilde{\mathbb{E}}[S_g(t)C(t)])r_d \\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[C(t)S_g(t)] &= -\min(\widetilde{\mathbb{E}}[C(t)S_g(t)], \widetilde{\mathbb{E}}[S(t)S_g(t)])r_r + \min(\widetilde{\mathbb{E}}[C(t)^2], \widetilde{\mathbb{E}}[S(t)C(t)])r_r - \min(\widetilde{\mathbb{E}}[C(t)], \widetilde{\mathbb{E}}[S(t)])r_r \\
&\quad + \widetilde{\mathbb{E}}[C_t(t)S_g(t)]r_t - \min(\widetilde{\mathbb{E}}[C_w(t)C(t)], \widetilde{\mathbb{E}}[S_g(t)C(t)])r_d \\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[C(t)S_b(t)] &= -\min(\widetilde{\mathbb{E}}[C(t)S_b(t)], \widetilde{\mathbb{E}}[S(t)S_b(t)])r_r + \widetilde{\mathbb{E}}[C_t(t)S_b(t)]r_t - \widetilde{\mathbb{E}}[C(t)S_b(t)]r_s + \widetilde{\mathbb{E}}[C(t)S(t)]r_b \\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[C_w(t)^2] &= 2\min(\widetilde{\mathbb{E}}[C_w(t)C(t)], \widetilde{\mathbb{E}}[C_w(t)S(t)])r_r + \min(\widetilde{\mathbb{E}}[C(t)], \widetilde{\mathbb{E}}[S(t)])r_r \\
&\quad - 2\min(\widetilde{\mathbb{E}}[C_w(t)^2], \widetilde{\mathbb{E}}[S_g(t)C_w(t)])r_d + \min(\widetilde{\mathbb{E}}[C_w(t)], \widetilde{\mathbb{E}}[S_g(t)])r_d \\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[C_w(t)C_t(t)] &= \min(\widetilde{\mathbb{E}}[C(t)C_t(t)], \widetilde{\mathbb{E}}[S(t)C_t(t)])r_r - \min(\widetilde{\mathbb{E}}[C_t(t)C_w(t)], \widetilde{\mathbb{E}}[C_t(t)S_g(t)])r_d
\end{aligned}
$$

$$
\begin{aligned}
&+ \min(\widetilde{\mathbb{E}}[C_w(t)^2], \widetilde{\mathbb{E}}[C_w(t)S_g(t)])r_d - \min(\widetilde{\mathbb{E}}[C_w(t)], \widetilde{\mathbb{E}}[S_g(t)])r_d - r_t\widetilde{\mathbb{E}}[C_w(t)C_t(t)] \\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[C_w(t)S(t)] &= \min(\widetilde{\mathbb{E}}[C(t)S(t)], \widetilde{\mathbb{E}}[S(t)^2])r_r - \min(\widetilde{\mathbb{E}}[C(t)C_w(t)], \widetilde{\mathbb{E}}[S(t)C_w(t)])r_r - \min(\widetilde{\mathbb{E}}[C(t)], \widetilde{\mathbb{E}}[S(t)])r_r \\
&\quad - \min(\widetilde{\mathbb{E}}[S(t)C_w(t)], \widetilde{\mathbb{E}}[S(t)S_g(t)])r_d + \min(\widetilde{\mathbb{E}}[C_w(t)^2], \widetilde{\mathbb{E}}[S_g(t)C_w(t)])r_d \\
&\quad - \min(\widetilde{\mathbb{E}}[C_w(t)], \widetilde{\mathbb{E}}[S_g(t)])r_d + \widetilde{\mathbb{E}}[C_w(t)S_b(t)]r_s - \widetilde{\mathbb{E}}[C_w(t)S(t)]r_b \\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[C_w(t)S_g(t)] &= \min(\widetilde{\mathbb{E}}[C_w(t)C(t)], \widetilde{\mathbb{E}}[C_w(t)S(t)])r_r + \min(\widetilde{\mathbb{E}}[C(t)S_g(t)], \widetilde{\mathbb{E}}[S(t)S_g(t)])r_r + \min(\widetilde{\mathbb{E}}[C(t)], \widetilde{\mathbb{E}}[S(t)])r_r \\
&\quad - \min(\widetilde{\mathbb{E}}[C_w(t)^2], \widetilde{\mathbb{E}}[C_w(t)S_g(t)])r_d - \min(\widetilde{\mathbb{E}}[C_w(t)S_g(t)], \widetilde{\mathbb{E}}[S_g(t)^2])r_d \\
&\quad + \min(\widetilde{\mathbb{E}}[C_w(t)], \widetilde{\mathbb{E}}[S_g(t)])r_d \\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[C_w(t)S_b(t)] &= \min(\widetilde{\mathbb{E}}[C(t)S_b(t)], \widetilde{\mathbb{E}}[S(t)S_b(t)])r_r - \min(\widetilde{\mathbb{E}}[C_w(t)S_b(t)], \widetilde{\mathbb{E}}[S_g(t)S_b(t)])r_d - \widetilde{\mathbb{E}}[C_w(t)S_b(t)]r_s \\
&\quad + \widetilde{\mathbb{E}}[C_w(t)S(t)]r_b \\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[C_t(t)^2] &= 2\min(\widetilde{\mathbb{E}}[C_t(t)C_w(t)], \widetilde{\mathbb{E}}[C_t(t)S_g(t)])r_d + \min(\widetilde{\mathbb{E}}[C_w(t)], \widetilde{\mathbb{E}}[S_g(t)])r_d - 2r_t\widetilde{\mathbb{E}}[C_t(t)^2] + r_t\widetilde{\mathbb{E}}[C_t(t)] \\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[C_t(t)S(t)] &= -\min(\widetilde{\mathbb{E}}[C(t)C_t(t)], \widetilde{\mathbb{E}}[S(t)C_t(t)])r_r + \min(\widetilde{\mathbb{E}}[C_w(t)C_t(t)], \widetilde{\mathbb{E}}[S_g(t)C_t(t)])r_d \\
&\quad + \min(\widetilde{\mathbb{E}}[S(t)C_w(t)], \widetilde{\mathbb{E}}[S_g(t)S(t)])r_d + \min(\widetilde{\mathbb{E}}[C_w(t)], \widetilde{\mathbb{E}}[S_g(t)])r_d \\
&\quad + \widetilde{\mathbb{E}}[S_b(t)C_t(t)]r_s - \widetilde{\mathbb{E}}[S(t)C_t(t)]r_b - r_t\widetilde{\mathbb{E}}[S(t)C_t(t)] \\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[C_t(t)S_g(t)] &= -\min(\widetilde{\mathbb{E}}[C_t(t)C_w(t)], \widetilde{\mathbb{E}}[C_t(t)S_g(t)])r_d + \min(\widetilde{\mathbb{E}}[C_w(t)S_g(t)], \widetilde{\mathbb{E}}[S_g(t)^2])r_d \\
&\quad - \min(\widetilde{\mathbb{E}}[C_w(t)], \widetilde{\mathbb{E}}[S_g(t)])r_d - r_t\widetilde{\mathbb{E}}[C_t(t)S_g(t)] + \min(\widetilde{\mathbb{E}}[C(t)C_t(t)], \widetilde{\mathbb{E}}[S(t)C_t(t)])r_r \\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[C_t(t)S_b(t)] &= \min(\widetilde{\mathbb{E}}[S_b(t)C_w(t)], \widetilde{\mathbb{E}}[S_b(t)S_g(t)])r_d - r_t\widetilde{\mathbb{E}}[C_t(t)S_b(t)] - \widetilde{\mathbb{E}}[C_t(t)S_b(t)]r_s + \widetilde{\mathbb{E}}[C_t(t)S(t)]r_b \\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[S(t)^2] &= -2\min(\widetilde{\mathbb{E}}[C(t)S(t)], \widetilde{\mathbb{E}}[S(t)^2])r_r + \min(\widetilde{\mathbb{E}}[C(t)], \widetilde{\mathbb{E}}[S(t)])r_r \\
&\quad + 2\min(\widetilde{\mathbb{E}}[C_w(t)S(t)], \widetilde{\mathbb{E}}[S(t)S_g(t)])r_d + \min(\widetilde{\mathbb{E}}[C_w(t)], \widetilde{\mathbb{E}}[S_g(t)])r_d + 2\widetilde{\mathbb{E}}[S_b(t)S(t)]r_s \\
&\quad + \widetilde{\mathbb{E}}[S_b(t)]r_s - 2\widetilde{\mathbb{E}}[S(t)^2]r_b + \widetilde{\mathbb{E}}[S(t)]r_b \\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[S(t)S_g(t)] &= \min(\widetilde{\mathbb{E}}[C(t)S(t)], \widetilde{\mathbb{E}}[S(t)^2])r_r - \min(\widetilde{\mathbb{E}}[C(t)S_g(t)], \widetilde{\mathbb{E}}[S(t)S_g(t)])r_r - \min(\widetilde{\mathbb{E}}[C(t)], \widetilde{\mathbb{E}}[S(t)])r_r \\
&\quad - \min(\widetilde{\mathbb{E}}[C_w(t)S(t)], \widetilde{\mathbb{E}}[S_g(t)S(t)])r_d + \min(\widetilde{\mathbb{E}}[C_w(t)S_g(t)], \widetilde{\mathbb{E}}[S_g(t)^2])r_d \\
&\quad - \min(\widetilde{\mathbb{E}}[C_w(t)], \widetilde{\mathbb{E}}[S_g(t)])r_d + \widetilde{\mathbb{E}}[S_b(t)S_g(t)]r_s - \widetilde{\mathbb{E}}[S_g(t)S(t)]r_b \\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[S(t)S_b(t)] &= -\min(\widetilde{\mathbb{E}}[C(t)S_b(t)], \widetilde{\mathbb{E}}[S(t)S_b(t)])r_r + \min(\widetilde{\mathbb{E}}[C_w(t)S_b(t)], \widetilde{\mathbb{E}}[S_g(t)S_b(t)])r_d \\
&\quad - \widetilde{\mathbb{E}}[S(t)S_b(t)]r_s + \widetilde{\mathbb{E}}[S_b(t)^2]r_s - \widetilde{\mathbb{E}}[S_b(t)]r_s - \widetilde{\mathbb{E}}[S_b(t)S(t)]r_b + \widetilde{\mathbb{E}}[S(t)^2]r_b - \widetilde{\mathbb{E}}[S(t)]r_b \\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[S_g(t)^2] &= 2\min(\widetilde{\mathbb{E}}[C(t)S_g(t)], \widetilde{\mathbb{E}}[S(t)S_g(t)])r_r + \min(\widetilde{\mathbb{E}}[C(t)], \widetilde{\mathbb{E}}[S(t)])r_r \\
&\quad - 2\min(\widetilde{\mathbb{E}}[C_w(t)S_g(t)], \widetilde{\mathbb{E}}[S_g(t)^2])r_d + \min(\widetilde{\mathbb{E}}[C_w(t)], \widetilde{\mathbb{E}}[S_g(t)])r_d \\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[S_g(t)S_b(t)] &= \min(\widetilde{\mathbb{E}}[C(t)S_b(t)], \widetilde{\mathbb{E}}[S(t)S_b(t)])r_r - \min(\widetilde{\mathbb{E}}[C_w(t)S_b(t)], \widetilde{\mathbb{E}}[S_g(t)S_b(t)])r_d \\
&\quad - \widetilde{\mathbb{E}}[S_g(t)S_b(t)]r_s + \widetilde{\mathbb{E}}[S(t)S_g(t)]r_b \\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[S_b(t)^2] &= -2\widetilde{\mathbb{E}}[S_b(t)^2]r_s + \widetilde{\mathbb{E}}[S_b(t)]r_s + 2\widetilde{\mathbb{E}}[S(t)S_b(t)]r_{break}
\end{aligned}
$$

Appendix B

Chapter 5

B.1 ODE systems

B.1.1 Second-order accumulated moments

In addition to the ODEs in Section A.1.2, the following ODEs are needed to calculate $\mathrm{Var}[\overline{S}(t)]$:

$$
\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[\overline{S}(t)^2] &= 2\cdot\widetilde{\mathbb{E}}[S(t)\overline{S}(t)] \\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[C(t)\overline{S}(t)] &= -\min(\widetilde{\mathbb{E}}[C(t)\overline{S}(t)],\widetilde{\mathbb{E}}[S(t)\overline{S}(t)])r_r + \widetilde{\mathbb{E}}[C_t(t)\overline{S}(t)]r_t + \widetilde{\mathbb{E}}[C(t)S(t)] \\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[C_w(t)\overline{S}(t)] &= \min(\widetilde{\mathbb{E}}[C(t)\overline{S}(t)],\widetilde{\mathbb{E}}[S(t)\overline{S}(t)])r_r - \min(\widetilde{\mathbb{E}}[C_w(t)\overline{S}(t)],\widetilde{\mathbb{E}}[S_g(t)]\overline{S}(t))r_d + \widetilde{\mathbb{E}}[C_w(t)S(t)] \\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[C_t(t)\overline{S}(t)] &= \min(\widetilde{\mathbb{E}}[C_w(t)\overline{S}(t)],\widetilde{\mathbb{E}}[S_g(t)\overline{S}(t)])r_d - r_t\widetilde{\mathbb{E}}[C_t(t)\overline{S}(t)] + \widetilde{\mathbb{E}}[C_t(t)S(t)] \\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[S(t)\overline{S}(t)] &= -\min(\widetilde{\mathbb{E}}[C(t)\overline{S}(t)],\widetilde{\mathbb{E}}[S(t)\overline{S}(t)])r_r + \min(\widetilde{\mathbb{E}}[C_w(t)\overline{S}(t)],\widetilde{\mathbb{E}}[S_g(t)\overline{S}(t)])r_d + \widetilde{\mathbb{E}}[S_b(t)\overline{S}(t)]r_s \\
&\quad - \widetilde{\mathbb{E}}[S(t)\overline{S}(t)]r_b + \widetilde{\mathbb{E}}[S(t)^2] \\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[S_g(t)\overline{S}(t)] &= \min(\widetilde{\mathbb{E}}[C(t)\overline{S}(t)],\widetilde{\mathbb{E}}[S(t)\overline{S}(t)])r_r - \min(\widetilde{\mathbb{E}}[C_w(t)\overline{S}(t)],\widetilde{\mathbb{E}}[S_g(t)\overline{S}(t)])r_d + \widetilde{\mathbb{E}}[S_g(t)S(t)] \\
\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\mathbb{E}}[S_b(t)\overline{S}(t)] &= -\widetilde{\mathbb{E}}[S_b(t)\overline{S}(t)]r_s + \widetilde{\mathbb{E}}[S(t)\overline{S}(t)]r_b + \widetilde{\mathbb{E}}[S_b(t)S(t)]
\end{aligned}
$$

B.2 Proofs

Proof of Theorem 7. For convenience let

$$h'(\boldsymbol{X}(t)) = \prod_{j=1}^{n} \overline{h_j(\boldsymbol{X})}(t)^{e_j}.$$

We have

$$
\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[h_0(\boldsymbol{X}(t))h'(\boldsymbol{X}(t))] \\
&= \lim_{h\to 0}\frac{1}{h}(\mathbb{E}[h_0(\boldsymbol{X}(t+h))h'(\boldsymbol{X}(t+h))] - \mathbb{E}[h_0(\boldsymbol{X}(t))h'(\boldsymbol{X}(t))]) \\
&= \lim_{h\to 0}\frac{1}{h}(\mathbb{E}[(h_0(\boldsymbol{X}(t+h)) - h_0(\boldsymbol{X}(t)))h'(\boldsymbol{X}(t))] \\
&\qquad + \mathbb{E}[h_0(\boldsymbol{X}(t+h))(h'(\boldsymbol{X}(t+h)) - h'(\boldsymbol{X}(t)))]) \\
&= \frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[h_0(\boldsymbol{X}(t))h'(\boldsymbol{X}(s))]|_{s=t} + \sum_{k=1}^{n}\mathbb{E}\left[\frac{\partial h}{\partial \overline{h_k(\boldsymbol{X})}}(\boldsymbol{X}(t))\right]
\end{aligned}
$$

where the second term is obtained by applying the bounded convergence theorem. For the first term

$$\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[h_0(\boldsymbol{X}(t))h'(\boldsymbol{X}(s))] \\
&= \frac{\mathrm{d}}{\mathrm{d}t}\int_r \sum_k rk\mathbb{P}(h_0(\boldsymbol{X}(t)) = k, h'(\boldsymbol{X}(s)) = r)\mathrm{d}r \\
&= \int_r \sum_k rk\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{P}(h_0(\boldsymbol{X}(t)) = k \mid h'(\boldsymbol{X}(s)) = r)\,\mathbb{P}(h'(\boldsymbol{X}(s)) = r)\mathrm{d}r \\
&= \int_r r\mathbb{E}[f_{X_i}(\boldsymbol{X}(t)) \mid h'(\boldsymbol{X}(s)) = r]\,\mathbb{P}(h'(\boldsymbol{X}(s)) = r) \\
&= \mathbb{E}[f_{h_0}(\boldsymbol{X}(t))h'(\boldsymbol{X}(s))]
\end{aligned}$$

and the result follows. □

Appendix C

Chapter 6

C.1 Proofs

Proof of Theorem 8. The joint process $(\boldsymbol{X}(t), \boldsymbol{\mathcal{Y}}(t))$ is clearly Markovian with infinitesimal generator $\mathcal{A}$ defined on h:

$$\begin{aligned}\mathcal{A}h(\boldsymbol{x},\boldsymbol{y}) &:= \lim_{\delta\to 0}\frac{\mathbb{E}[h(\boldsymbol{X}(t+\delta),\boldsymbol{\mathcal{Y}}(t+\delta))|(\boldsymbol{X}(t),\boldsymbol{\mathcal{Y}}(t))=(\boldsymbol{x},\boldsymbol{y})]-h(\boldsymbol{x},\boldsymbol{y})}{\delta}\\ &= \sum_{i=1}^{M} g_i(\boldsymbol{x},\boldsymbol{y})\frac{\partial h}{\partial y_i}(\boldsymbol{x},\boldsymbol{y}) + \sum_{c\in\mathcal{C}} r_c(\boldsymbol{x},\boldsymbol{y})[h(\boldsymbol{x}+\boldsymbol{\delta}_c,\boldsymbol{y})-h(\boldsymbol{x},\boldsymbol{y})]\end{aligned}$$

It thus follows by Dynkin's formula [e.g. 116, Lemma 17.21] that for $t\in\mathbb{R}_+$:

$$\mathbb{E}[h(\boldsymbol{X}(t),\boldsymbol{Y}(t))] = h(\boldsymbol{x}_0,\boldsymbol{y}_0) + \int_0^t \mathbb{E}[\mathcal{A}h(\boldsymbol{X}(s),\boldsymbol{\mathcal{Y}}(s))]\,\mathrm{d}s$$

□

Proof of Theorem 9. This proof is originally due to Hayden [14]. We begin by representing each process $(\bar{\boldsymbol{X}}^S(t),\bar{\boldsymbol{Y}}^S(t))$ in terms of mutually independent rate-1 Poisson processes $\{P_c(t): c\in\mathcal{C}\}$ by the *random-time change* approach [72]:

$$\begin{aligned}\bar{\boldsymbol{X}}^S(t) &= \boldsymbol{x_0} + \sum_{c\in\mathcal{C}} P_c\left(\int_0^t r_c^S(\boldsymbol{X}^S(s),\boldsymbol{\mathcal{Y}}^S(s))\,\mathrm{d}s\right)\boldsymbol{\delta}_c/S\\ \bar{\boldsymbol{\mathcal{Y}}}^S(t) &= \boldsymbol{y_0} + \int_0^t \boldsymbol{g}(\bar{\boldsymbol{X}}^S(s),\bar{\boldsymbol{\mathcal{Y}}}^S(s))\,\mathrm{d}s\end{aligned}$$

On S, $\boldsymbol{f}(\cdot,\cdot):\mathbb{R}^n\times\mathbb{R}^m\to\mathbb{R}^n$ and $\boldsymbol{g}(\cdot,\cdot):\mathbb{R}^n\times\mathbb{R}^m\to\mathbb{R}^m$ are both Lipschitz continuous uniformly; let K be a Lipschitz constant for both functions. Now define:

$$D^S(t) := \sup_{s\in[0,t]}\left\|\bar{\boldsymbol{X}}^S(s) - \boldsymbol{x}_0 - \int_0^s \boldsymbol{f}(\bar{\boldsymbol{X}}^S(u),\bar{\boldsymbol{\mathcal{Y}}}^S(u))\,\mathrm{d}u\right\|$$

and $\epsilon^S(t) := \|\bar{\boldsymbol{X}}^S(t)-\boldsymbol{x}(t)\| + \|\bar{\boldsymbol{Y}}^S(t)-\boldsymbol{y}(t)\|$. Then we have for $t\in[0,t_f]$:

$$\begin{aligned}\epsilon^S(t) &\le D^S(T) + \int_0^t \|\boldsymbol{g}(\bar{\boldsymbol{X}}^S(s),\bar{\boldsymbol{\mathcal{Y}}}^S(s)) - \boldsymbol{g}(\boldsymbol{x}(s),\boldsymbol{y}(s))\|\,\mathrm{d}s\\ &+ \int_0^t \|\boldsymbol{f}(\bar{\boldsymbol{X}}^S(s),\bar{\boldsymbol{\mathcal{Y}}}^S(s)) - \boldsymbol{f}(\boldsymbol{x}(s),\boldsymbol{y}(s))\|\,\mathrm{d}s \;\le D^S(t_f) + 2K\int_0^t \epsilon^S(s)\,\mathrm{d}s\end{aligned}$$

and by Grönwall's inequality, we obtain $\epsilon^S(t) \leq D^S(T)\exp(2Kt_f)$. Now note that:

$$D^S(T) \leq \sup_{s\in[0,t_f]} \left\| \sum_{c\in\mathcal{C}} \frac{\boldsymbol{\delta}_c}{S} \left[P_c\left(\int_0^s r_c^S(\boldsymbol{X}^S(u), \boldsymbol{\mathcal{Y}}^S(u))\,\mathrm{d}u \right) - \int_0^s r_c^S(\boldsymbol{X}^S(u), \boldsymbol{\mathcal{Y}}^S(u))\,\mathrm{d}u \right] \right\|$$

which can be bounded above by $\sum_{c\in\mathcal{C}} \|\boldsymbol{\delta}_c\| \sup_{s\in[0,t_f]} |P_c(SCs)/S - Cs|$ for some $C \in \mathbb{R}_+$ independent of S. The result then follows by the strong law of large numbers for the Poisson process, which is equivalent to the functional strong law of large numbers [e.g. 190, Section 3.2], that is, for all $s \in \mathbb{R}_+$, $\sup_{u\in[0,s]} \|P_c(Su)/S - u\| \to 0$ as $S \to \infty$ with probability 1. □

Proof of Theorem 10. This proof is originally due to Hayden [14]. We assume the representation of the processes $\bar{\boldsymbol{X}}^S(t)$ and $\bar{\boldsymbol{\mathcal{Y}}}^S(t))$ given in Equation C.1. Further it is possible [72, Corollary 5.5 and Remark 5.4] to construct, on the same probability space as these processes, mutually independent standard Brownian motions $\{B_c(t) : c \in \mathcal{C}\}$, such that:

$$Z_c := \sup_{t\in\mathbb{R}_+} \frac{|P_c(t) - t - B_c(t)|}{\log(2 \vee t)} < \infty \quad \text{almost surely}$$

From this it follows that as $N \to \infty$, almost surely:

$$\sqrt{S} \sup_{t\in[0,t_f]} \left\| \bar{\boldsymbol{X}}^S(t) - \boldsymbol{x}_0 - \int_0^t \boldsymbol{f}(\bar{\boldsymbol{X}}^S(s), \bar{\boldsymbol{\mathcal{Y}}}^S(s))\,\mathrm{d}s - \sum_{c\in\mathcal{C}} B_c\left(\int_0^t r_c^S(\boldsymbol{X}^S(s), \boldsymbol{\mathcal{Y}}^S(s))\,\mathrm{d}s \right)(\boldsymbol{\delta}_c/S) \right\| \longrightarrow 0 \tag{C.1}$$

A direct comparison of $\frac{\boldsymbol{X}^S(t) - N\boldsymbol{x}(t)}{\sqrt{S}}$ with $\boldsymbol{E}^X(t)$ and similarly for $\boldsymbol{E}^Y(t)$ using Equation C.1 yields the result. We omit further details here for the sake of brevity. □

Appendix D

Chapter 8

D.1 GPA Syntax

System

$$
\begin{aligned}
System &:= Parameters\ \ VariableDef^{*}\, ModelDef\ \ Continuous Variables \\
&\quad\ Analysis^{*}\ \ Experiment^{*} \\
Parameters &:= ParameterDef^{*}\ \ TDParameterDef^{*} \\
ParameterDef &:= parameterId\ \texttt{=}\ number\texttt{;} \\
TDParameterDef &:= \texttt{load "}filename\texttt{"}integer\ \texttt{into}\ parameterId\texttt{;} \\
VariableDef &:= Variable\texttt{=}Expression(NonMoment) \mid Expression(Moment) \\
Variable &:= \texttt{\$}variableId \\
Expression(x) &:= Expression(x)(\texttt{+} \mid \texttt{-} \mid \texttt{*} \mid \texttt{/} \mid \texttt{\^{}})Expression(x) \\
&\quad\ functionID\texttt{(}Expression(x)(\texttt{,}Expression(x))^{+}\texttt{)} \mid Terminal(x) \\
Terminal(Constant) &:= realnumber \mid integer \mid parameter \mid \texttt{t} \mid Variable \\
Terminal(NonMoment) &:= Terminal(Constant) \mid \Big((Population \mid Continuous Variable)(\texttt{\^{}}integer)^{?}\Big)^{+} \\
Terminal(Moment) &:= Terminal(Constant) \mid \Big(\texttt{E} \mid \texttt{Var} \mid \texttt{Skew}\Big)\ \texttt{[}LinCombination(NonMoment)\texttt{]} \\
&\quad\ \texttt{Cov[}LinCombination(NonMoment)\texttt{,}LinCombination(NonMoment)\texttt{]} \\
&\quad\ \texttt{Moment[}LinCombination(NonMoment)\texttt{,}integer\texttt{]} \\
&\quad\ \texttt{Eg[}Expression(NonMoment)\texttt{]}
\end{aligned}
$$

Plain PCTMC models

$$
\begin{aligned}
Population &:= \texttt{\{}populationID\texttt{\}} \\
ModelDef &:= TransitionClass^{*}\ \ InitValues
\end{aligned}
$$

$$TransitionClass := PopulationList \texttt{ -> } PopulationList \texttt{ @} Expression(NonMoment)\texttt{;}$$

$$PopulationList := Population \left(\texttt{+} Population\right)^*$$

$$InitValues := \left(Population\texttt{=}Expression(Constant)\texttt{;}\right)^*$$

GPEPA models

$$Population := groupLabel\texttt{:}Agent \mid \texttt{\#}actionId$$

$$ModelDef := AgentDef^* \; SystemEquation \; ActionCount^?$$

$$AgentDef := agentID \texttt{ = } Agent$$

$$Agent := Agent\texttt{<}ActionList^?\texttt{>}Agent$$

$$\mid Summation \mid agentId \mid \texttt{(}Agent\texttt{)}$$

$$Summation := Prefix(\texttt{+}Prefix)^*$$

$$Prefix := \texttt{(}actionId\texttt{, }parameterId\texttt{).(}\texttt{(}Summation\texttt{)} \mid \texttt{stop} \mid agentId \mid\texttt{);}$$

$$SystemEquation := \texttt{(}SystemEquation\texttt{<}ActionList^?\texttt{>}SystemEquation\texttt{)}$$

$$\mid groupLabel\texttt{\{}AgentsParallel\texttt{\}}$$

$$AgentsParallel := FluidAgent(\texttt{ |}FluidAgent)^*$$

$$FluidAgent := agentId \; (\texttt{[}Expression(Constant)\texttt{]})^?$$

$$ActionList := actionId \; (\texttt{, } actionId)^*$$

$$ActionCount := \texttt{Count } actionId \left(\texttt{,}actionId\right)^* \texttt{;}$$

Continuous variables

$$ContinuousVariables := ContinuousDef^* \; ContinuousInit^*$$

$$ContinuousDef := \texttt{ddt } \texttt{~}contId \texttt{ = } Expression(NonMoment)\texttt{;}$$

$$ContinuousInit := \texttt{~}contId \texttt{ = } Expression(Constant)\texttt{;}$$

$$ContinuousVariable := \texttt{~}contId \mid \texttt{acc(}Terminal(NonMoment)\texttt{)}$$

Analyses

$$Analysis := ODEs \mid Simulation$$

$$ODEs := \texttt{ODEs} \left(\texttt{[}Options\texttt{]}\right)^? \texttt{ (stopTime = } realnumber\texttt{, stepSize = } realnumber\texttt{,}Options^?\texttt{)}$$

$$\texttt{\{}Plot^{*}\texttt{\}}$$

$$Simulation := \texttt{Simulation(stopTime =}\ realnumber\texttt{, stepSize =}\ realnumber\texttt{, replications =}\ integer\texttt{)}\ \texttt{\{}Plot^{*}\texttt{\}}$$

$$Options := optionId\ \texttt{=}\ optionValue\ \left(\texttt{,}\ optionId\ \texttt{=}\ optionValue\right)^{*}$$

$$Plot := Expression(Moment)\ \left(\texttt{,}\ Expression(Moment)\right)^{*}\ \left(\texttt{-> "}filename\texttt{"}\right)^{?}\ \texttt{;}$$

Experiments

$$Experiment := Iterate \mid Distribution$$

$$Iterate := \left(\texttt{Iterate}\ Range^{+}\right)^{?}\ \left(\texttt{Minimise}\ PlotAt\ Range^{+}\ \left(\texttt{where}\ ParameterDef^{+}\right)^{?}\right)^{?}\ Analysis\ \texttt{plot \{}\ \left(PlotAt\texttt{;}\right)^{+}\ \texttt{\}}$$

$$Range := parameterId\ \texttt{from}\ realnumer\ \texttt{to}\ realnumber\ Step$$

$$Step := \left(\texttt{in}\ integer\ \texttt{steps}\right) \mid \left(\texttt{with step}\ realnumber\right)$$

$$PlotAt := ExpressionAt\ \left(\texttt{when}\ ExpressionAt\texttt{>=}realnumber\ \left(\texttt{and}\ ExpressionAt\ \texttt{>=}\ realnumber\right)^{*}\right)^{?}$$

$$ExpressionAt := Expression(Moment)\ \texttt{at}\ realnumber$$

Appendix E

GPEPAc model of HTCondor

User agents

Each different environment $i \in \mathcal{C}_{PC}$ (such as library, labs, etc.) is subject to arrivals and departures of interactive users. A user agent arrives into the environment according to an external arrival process, generating an $arrive_i$ action:

$$Arrive\_user_i \stackrel{def}{=} (arrive_i, r_{arrive,i}(t)).Arrive\_user_i$$

Each agent logs into a randomly chosen workstation and logs out after a period of time:

$$\begin{aligned} User\_ready_i &\stackrel{def}{=} (arrive_i, \top).User \\ User_i &\stackrel{def}{=} (login \text{ get } \underline{pc}, r_{login,i}).User_i^{\underline{pc}} \\ User_i^{\underline{pc}} &\stackrel{def}{=} (logout \text{ send on } \underline{pc}, r_{logout,i}i(t)).User\_ready_i \end{aligned}$$

Job agents

Similarly, jobs from each different group $i \in \mathcal{C}_{Job}$ are submitted to the respective queue according to an external process generating an $submit_i$ action:

$$Submit\_job_i \stackrel{def}{=} (submit_i, r_{submit,i}(t)).Submit\_job_i$$

Each job agent can be removed from the queue to be assigned to a workstation. At the workstation, the job is processed until finished or evicted by an incoming user, returning back to the queue:

$$\begin{aligned} Job\_ready_i &\stackrel{def}{=} (submit_i, \top).Job_i \\ Job_i &\stackrel{def}{=} (dequeue \text{ get } \underline{pc}, r_{submit,i}).Job_i^{\underline{pc}} \\ Job_i^{\underline{pc}} &\stackrel{def}{=} (process \text{ send on } \underline{pc}, r_{proc,i}).Job\_done_i + (evict \text{ receive on } \underline{pc}, \top).Job_i \end{aligned}$$

In this example, the job is processed in a single phase with an exponentially distributed duration. However, in real applications, job durations come from a range of distributions. For example, the job could be executed in a sequence of phases, for example, a database retrieval followed by a numerical computation. The job could terminate at each phase. If the durations at each phase are exponentially distributed, the distribution of the overall duration belongs to the family of *Coxian distributions*. These can be used to accurately approximate a wide range of distributions. The GPEPAc framework can directly model multi-phase communication with a Coxian structure. For example, the job agent can be processed in three phases instead of one:

$$\begin{aligned} Job_i^{\underline{pc}} \stackrel{def}{=} &(end \text{ send on } \underline{pc}, r_{proc,i,0}).Job\_done_i + (process \text{ send on } \underline{pc}, r_{proc,i,0,1}).Job_{i,1} \\ &(evict \text{ receive on } \underline{pc}, \top).Job_i \end{aligned}$$

$$Job_{i,1}^{\underline{pc}} \stackrel{def}{=} (end \text{ send on } \underline{pc}, r_{proc,i,1}).Job\_done_i + (process \text{ send on } \underline{pc}, r_{proc,i,1,2}).Job_{i,2}$$
$$(evict \text{ receive on } \underline{pc}, \top).Job_i$$
$$Job_{i,2}^{\underline{pc}} \stackrel{def}{=} (end \text{ send on } \underline{pc}, r_{proc,i,2}).Job\_done_i + (evict \text{ receive on } \underline{pc}, \top).Job_i$$

Similar modification can be applied to model more general inter-arrival and inter-submission times for users and jobs respectively and interactive usage durations.

Workstations and HTCondor

Each workstation from an environment $i \in \mathcal{C}_{PC}$ is available to interactive users and a workstation with an active user waits until the user logs out. Workstations without users become available to HTCondor jobs after an idle period. In such case it expects a job to be assigned (by a scheduler) for processing. While processing a job, the workstation is also available to users to log in, in which case it evicts the currently processed job:

$$PC_i \stackrel{def}{=} (login \text{ init } \underline{u}, r_{login,i}).PC_i^{\underline{u}} + (avail, r_{avail,i}).PC\_avail_i$$
$$PC_i^{\underline{u}} \stackrel{def}{=} (logout \text{ receive on } \underline{u}, r_{logout,i}).PC_i$$
$$PC\_avail_i \stackrel{def}{=} (assign_i \text{ get } \underline{j}).PC_i^{\underline{j}} + (login \text{ init } \underline{u}, r_{login,i}).PC_i^{\underline{u}}$$
$$PC_i^{\underline{j}} \stackrel{def}{=} (process \text{ receive on } \underline{j}, r_{process,i}).PC_i^{\underline{j}} + (end \text{ receive on } \underline{j}, r_{process,i}).PC\_avail_i$$
$$(login \text{ init } \underline{u}, r_{login,i}).PC\_evict_i^{\underline{j},\underline{u}}$$
$$PC\_evict_i^{\underline{j},\underline{u}} \stackrel{def}{=} (evict \text{ send on } \underline{j}, r_{evict,i}).PC_i^{\underline{u}}$$

The core HTCondor system is captured by scheduler agents. Each scheduler provides an interface between jobs from the different high-throughput groups and different workstation classes. It dequeues jobs from the respective queues and assigns them to workstations.

$$Sch \stackrel{def}{=} \sum_{i \in \mathcal{C}_{Job}} (dequeue_i \text{ init } \underline{j}, r_{dequeue,i}).Sch_i^{\underline{pc}}$$
$$Sch_i^{\underline{pc}} \stackrel{def}{=} \sum_{j \in \mathcal{C}_{PC}} (assign_j, r_{assign} \text{ forward } \underline{j}, i, j).Sch$$

System equation

We present a simple example consisting of only one type workstations and users – A for "all" users and workstations from the campus. We consider two job groups – B for the "background" jobs submitted by all the different high-throughput groups, and H for "hypothetical" jobs that we will use to evaluate the system under various conditions.

The whole system can be described by the system equation below:

$$\Big(\mathbf{Queue}\{\|_{i\in\{B,H\}}\ Job\_ready_i[\infty]\} \underset{\{submit_i, i\in\{B,H\}\}}{\bowtie} \mathbf{SubmissionsJ}\{\|_{i\in\{B,H\}}\ Submit\_job_i\}\Big)$$
$$\underset{\{dequeue_i, i\in\{B,H\}\}}{\bowtie} \mathbf{Schedulers}\{Sch[n_{Sch}]\} \underset{\{assign_A\}}{\bowtie}$$
$$\Big(\mathbf{PCs}\{PC_A[n_{PC,A}]\} \underset{\{login_A\}}{\bowtie} \mathbf{Users}\{User_A[\infty]\} \underset{\{arrive_A\}}{\bowtie} \mathbf{ArrivalsU}\{Arrive\_user_A\}\Big)$$

We use infinite populations here to model open arrivals of users and jobs. This results in a valid PCTMC, because the infinite populations always occur inside the minimum operator where the other argument is finite.