Metaheuristics in Stochastic Combinatorial Optimization: a Survey

Leonora Bianchi*, Marco Dorigo, Luca Maria Gambardella, and Walter J. Gutjahr

March 24, 2006

Abstract

Metaheuristics such as Ant Colony Optimization, Evolutionary Computation, Simulated Annealing, Tabu Search and Stochastic Partitioning Methods are introduced, and their recent applications to a wide class of combinatorial optimization problems under uncertainty are reviewed. The flexibility of metaheuristics in being adapted to different modeling approaches and problem formulations emerges clearly. This paper provides also a description and classification of the modeling approaches of optimization under uncertainty. Moreover, a formal description of the main formulations corresponding to more classical domains in the literature is provided. In this survey, the reader familiar to metaheuristics finds also pointers to classical algorithmic approaches to optimization under uncertainty, while the reader new to metaheuristics should find a good tutorial in those metaheuristics that are being applied to optimization under uncertainty.

1 Introduction

There is an increasing interest of the operations research community in addressing optimization problems that include in their mathematical formulation uncertain, stochastic, and dynamic information. Problem solving under uncertainty has a very high impact on real world contexts, since optimization problems arising in practice are becoming increasingly complex and dynamic, also thanks to the fast development of telecommunications that makes not only the perception but also the changes of the world more rapid, stochastic and difficult to forecast. The focus of this paper is on Stochastic Combinatorial Optimization Problems (SCOPs), a wide class of combinatorial optimization problems

*IDSIA - Dalle Molle Institute for Artificial Intelligence, Via Cantonale, Galleria 2, 6928 Manno, Switzerland. E-mail: leonora@idsia.ch

under uncertainty, where all or part of the information about the problem data is unknown, but some knowledge about its probability distribution is assumed.

In recent years, metaheuristic algorithms such as Ant Colony Optimization (ACO), Evolutionary Computation (EC), Simulated Annealing (SA), Tabu Search (TS), Stochastic Partitioning Methods (SPM), and others, are emerging as successful alternatives to classical approaches based on mathematical and dynamic programming for solving SCOPs. In fact, due to the high complexity and difficulty of optimization problems under uncertainty, often classical approaches (that guarantee to find the optimal solution) are feasible only for small size instances, and they could require a lot of computational effort. In contrast, approaches based on metaheuristics are capable of finding good and sometimes optimal solutions to problem instances of realistic size, in a generally smaller computation time. Table 1 lists some papers in the literature providing evidence about the advantages in solving SCOPs[1] via metaheuristics instead of using exact classical methods.

This survey paper is the first attempt to put under a unifying view the several applications of metaheuristics to SCOPs, and it should contribute in balancing the literature, where a number of surveys and books about solving SCOPs via classical techniques exist, but none about using metaheuristics, despite the research literature is already quite rich. The remainder of the paper is organized as follows. Section 2 proposes a classification of the modeling approaches to uncertainty according to dynamicity and type of description of uncertain data, and precisely defines the scope of SCOPs. Section 3 recalls the main formal definitions of both static and dynamic SCOPs from the literature, by providing links to the corresponding algorithmic domains, with the aim of giving a clear view of the intersection between classical approaches and new ones based on metaheuristics. This section also introduces the issue of objective function computation in SCOPs, which may involve different types of objective function approximations. Section 4 reviews the main applications to SCOPs of metaheuristics for which a significant amount of interesting literature exist, namely ACO, EC, SA, TS, SPM, Progressive Hedging (PH), and Rollout Algorithms (RO). Section 5 discusses some remarks drawn by taking a transversal view on the reviewed metaheuristics, and proposes possible directions of research. Finally, section 6 highlights the conclusions.

[1]Legend for the SCOPs of Table 1: VRPSD = vehicle routing problem with stochastic demands, SCP = set covering problem, TSPTW = traveling salesman problem with stochastic time windows, PTSP = probabilistic traveling salesman problem, SSP = shop scheduling problem, SDTCP = stochastic discrete time-cost problem, SOPTC = sequential ordering problem with time constraints, VRPSDC = vehicle routing problem with stochastic demands and customers.

Reference(s)	SCOP	Metaheuristic(s)	Exact method	Advantage of the metaheuristic(s) over the exact method
Beraldi and Ruszczyński [13] (2005)	SCP	SPM (Beam Search)	Branch and Bound	Maximum deviation from optimal solution is 5%, and in some cases Beam Search finds the optimal solution. The running time reduction with respect to Branch and Bound is roughly between 20% and 90%
Bianchi et al. [21] [22] (2005)	VRPSD	ACO, SA, TS, ILS, EC	Integer L-shaped method by Gendreau et al. [65] solves instances with up to 70 customers	ACO, SA, TS, ILS, EC address instances with up to 200 customers
Gutjahr [75] (2004)	TSPTW	ACO	Complete Enumeration solves in about 4 hours instances with 10 customers	ACO and SA solve instances with up to 20 customers in a few seconds
Branke and Guntsch [33] (2003) [34] (2004), Bowler et al. [32] (2003), Bianchi [24] [25] (2002)	PTSP	ACO, SA	Branch and Cut by Laporte et al. [100] solves instances with up to 50 customers	In [24, 25], ACO addresses instances with up to 200 customers. In [33, 34] ACO addresses instances with up to 400 customers. In [32], SA addresses instances with up to 120 customers
Finke et al. [56] (2002)	SSP	TS	Mixed Integer Linear Programming solves instances with up to 10 jobs and 5 machines	29 out of 30 small instances solved to optimality. Instances with up to 20 jobs and 10 machines have been addressed.
Gutjahr et al. [79] (2000)	SDTCP	SPM (Stochastic Branch and Bound)	Complete Enumeration approaches fail to generate results within a reasonable amount of time even for medium-size problems	Stochastic Branch and Bound outperforms classic techniques both in solution quality and in runtime
Costa and Silver [44] (1998)	SOPTC	TS	Branch and Bound solves instances with up to 14 causes, with a computation time from 0.1 to about 30000 seconds	TS is much faster (always 0.1 or 0.2 seconds for the small instances with up to 14 customers). Addressed also big instances with up to 500 customers
Gendreau et al. [66] (1996)	VRPSDC	TS	Integer L-shaped method by Gendreau et al. [65] solves instances with up to 70 customers	TS is faster (from Tables I and II of [66] the time gain of TS with respect to the exact method may be computed)

Table 1: Evidence about some of the advantages in solving SCOPs via metaheuristics instead of using exact classical methods.

2 Modeling approaches to uncertainty

In defining the scope of SCOPs one faces the difficulty of considering the many ways in which uncertainty may be formalized. Uncertainty is included in the formulation of optimization problems in order to go nearer to real world conditions, but models should also be a bit simplified, in order to be tractable analytically or numerically. The efforts done in reaching a good trade-off between usefulness of the model and tractability of the problem have produced a multitude of formalizations of uncertainty. This is even more evident for metaheuristics, because, due to their simplicity, they may be easily applied to complex formulations that would be considered intractable for many classical algorithmic approaches.

When considering models of optimization problems under uncertainty, there are mainly two aspects to define: first, the way uncertain information is formalized, and second, the dynamicity of the model, that is, the time uncertain information is revealed with respect to the time at which decisions must be taken. The several modeling approaches differ in the way the first and/or the second aspects are defined. Here, we propose a classification of models according to these two aspects, uncertainty and dynamicity, as schematized in Figure 1. For space limitations, this paper will then focus only on a subset of models, that correspond to our definition of SCOPs.

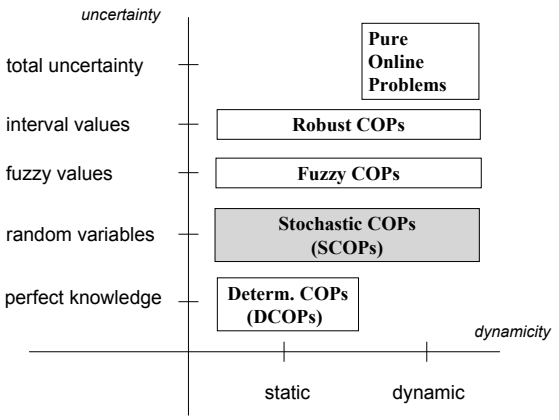

Figure 1: Scheme for the conceptual classification of Combinatorial Optimization Problems (COPs) under uncertainty. This paper focuses on Stochastic COPs (SCOPs) and to solution methods based on metaheuristics.

Uncertain information may be formalized in several ways (vertical axis of Figure 1). The case of perfect knowledge about the data of the problem corresponds to the classical field of solving (Deterministic) Combinatorial Optimization Problems (DCOPs) (low left corner of Figure 1). Here, all information is

available at the decision stage, and it is used by optimization algorithms to find a possibly optimal solution. The concrete application of a solution found would lead exactly to the cost of the solution as computed by the optimization algorithm, therefore DCOPs are also considered static problems, because from the point of view of the decision maker, there is nothing else to be decided after the optimization took place[2]. A typical example of DCOP is the well known Traveling Salesman Problem (TSP) [73], where, given a set of customers and the set of distance values among each couple of customers, one must find the Hamiltonian tour (that is, a tour visiting once each customer) of minimal length. Despite its simple formulation, the TSP is an NP-hard problem, like many DCOPs.

Let us now consider problem formulations involving uncertainty (upper levels of Figure 1). One possibility is to describe uncertain information by means of random variables of known probability distributions. This is what we assume in SCOPs (a more precise definition and examples of SCOPs will be given in section 3). Under this assumption, the optimization problem is stochastic, and the objective function strongly depends on the probabilistic structure of the model. Typically, the objective function involves quantities such as an expected cost, the probability of violation of some constraint, variance measures, and so on. In SCOPs one can distinguish a time *before* the actual realization of the random variables, and a time *after* the random variables are revealed, because the associated random events happen. Static SCOPs are characterized by the fact that decisions, or, equivalently, the identification of a possibly optimal solution, is done *before* the actual realization of the random variables. This framework is applicable when a given solution may be applied with no modifications (or very small ones) once the actual realization of the random variables are known. The literature sometimes addresses this type of problems as 'a-priori' optimization. As an example of this class of problems, consider the probabilistic TSP (PTSP), that consists in finding a Hamiltonian tour visiting all customers (the 'a priori' tour) of minimum expected cost, given that each customer has a known probability of requiring a visit. Once the information of which customers actually require a visit on a certain day is known, the customers requiring a visit are visited in the order of the 'a priori' tour, simply skipping the customers not requiring a visit.

Dynamic SCOPs arise when it is not possible or not convenient to design a solution that is usable as it is for any realization of the random variables. In this case, decisions that need an optimization effort must be taken also *after* the random events have happened. This could also be done in stages, because it is often the case that the uncertain information is not revealed all at once, but in stages. As an example of dynamic SCOP, consider for instance a TSP where new customers of known positions appear with a certain probability while the salesman has already started to visit the customers known a priori. In this case an a priori tour must be modified dynamically in order to include the new

[2]Nevertheless, a solution algorithm may use a 'dynamic' or 'stochastic' mechanism also in these cases, as, for example, the dynamic programming algorithm applied to (deterministic, static) shortest path problems, or algorithms that involve some random choice such as virtually all metaheuristics and local search procedures.

customers in the visiting tour.

Another way of formalizing uncertainty is to identify the uncertain information with fuzzy quantities (vectors or numbers), and constraints with fuzzy sets. This approach has its roots in Bellman and Zadeh [12] and in Zimmermann [144], but currently occupies a minor portion of the optimization literature.

An approach which is receiving increasing attention in the last years is the one of robust optimization, which assumes that uncertain information is known in the form of interval values. For example, one could consider the robust TSP, where the cost of arcs between couples of customers is given by interval values. These costs could have the meaning of travel times, being small if there is no or little traffic, and being high in case of traffic congestion. The robustness approach consists in finding solutions that hedge against the worst contingency that may arise, given that no knowledge about the probability distribution of random data is known. One possible way of quantifying robustness is the *min-max* criterion, under which the robust decision is that for which the highest level of cost taken across all possible future input data scenarios is as low as possible. Both static and dynamic versions of robust optimization problems may be formulated. For a good introduction to robust optimization, see for instance the book by Kouvelis and Yu [99].

On the highest level of Figure 1 we placed problems that we call Pure Online, where the input is modeled as a sequence of data which are supplied to the algorithm incrementally, but without making any assumption that can help to make a prevision on the new data. An algorithm for a Pure Online Problem produces the output incrementally without knowing the complete input, and its performance is evaluated with respect to an abstract competitor, who knows all the complete (past and future) data, and that is able to solve the offline problem optimally. This way of evaluating algorithms is called in the literature *competitive analysis* [3, 31]. An example of Pure Online problem is the Dynamic Traveling Repair Problem [91], where a set of servers move from point to point in a metric space; the speed of each server is constant, so the time it takes to travel from one point to another is proportional to the distance between two points; time is continuous and at any moment a request for service can arrive at any point in the space; each job also specifies a deadline; if a job is serviced, a server must reach the point where the request originated by its deadline; the goal is to service as many incoming request as possible by their deadlines.

We should again remark that in this paper we restrict to SCOPs (the shaded box in Figure 1). SCOPs are *combinatorial* optimization problems, that is, problems where the decision space is finite but possibly too big to be enumerated, and/or problems having a combinatorial structure because solutions are encoded by permutations, binary vectors or other combinatorial objects. By this choice we neglect the vast field of continuous optimization under uncertainty, although the scheme we have just proposed for classifying problems under uncertainty equally applies to continuous problems.

SCOPs are relevant in many practical contexts, such as vehicle routing problems, where stochasticity is due to variable customers demands, or variable travel times, routing on information networks, where stochasticity is due to

the variability of traffic and the related speed of information packages, finance, scheduling, location problems and many other contexts. All these problem domains may be, and usually are, also modeled as DCOPs. The advantage of using SCOPs over DCOPs is that the solutions produced may be more easily and better adapted to practical situations where uncertainty cannot be neglected, such as thrash collection, cash collection from banks, location of emergency services, and so on. Of course, the use of SCOPs instead of DCOPs comes at a price: first, the objective function is typically much more computationally demanding in SCOPs than in DCOPs; second, for a practical application of SCOPs, there is the need to assess probability distributions from real data or subjectively, a task that is far from trivial. For a discussion about the issue of computational burden and complexity in certain SCOP formulations, see for instance Haneveld and van der Vlerk [83], and Dyer and Stougie [52]. The ways this issue is managed in metaheuristics applied to SCOPs will be described in detail in section 4.

3 Formal descriptions of SCOPs

The class of SCOPs is so important and has impact in so many domains that several research areas are dedicated to its study: Stochastic Integer Programming, Markov Decision Processes (which is part of Stochastic Dynamic Programming) and Simulation Optimization being the main ones. Each research area corresponds to a particular way of modeling, formulating and solving optimization problems under uncertainty, and it is often treated separately in the optimization literature. The application of metaheuristics to SCOPs is a quite recent and fast growing research area, and it is thus natural that many of the papers borrow from the classical SCOP literature the same problem formulations. In this section we first give a general definition of a SCOP, then, in sections 3.1 and 3.2, we recall the main formal definitions of both static and dynamic SCOPs from the literature, by giving pointers to the research areas that originally proposed them. In section 3.3, we introduce the issue of objective function computation in SCOPs, which may involve different types of objective function approximations.

Let us now give a general definition of SCOP, as proposed by Kall and Wallace [96]. (Throughout the paper we use the minimization form for optimization problems, the maximization form is equivalent and can be derived in a straightforward manner by substituting the word 'min' with the word 'max').

Definition 1 (SCOP)
Consider a probability space (Ω, Σ, P) *([72]), where* Ω *is the domain of random variables* ω *(typically a subset of* $\mathbb{R}^k$*),* Σ *is a family of "events", that is subsets of* Ω*, and* P *is a probability distribution on* Σ*, with* $P(\Omega) = 1$*. Consider also a finite set* S *of decision variables* x*.* S *is typically a subset of* $\mathbb{R}^n$*. The random variable* ω *could also depend on the decision variable* x*, in that case it is denoted by* ω_x*. Given a cost function* G *and constraint functions* H_i*,* $i = 1, 2, \ldots, m$*,*

mapping $(x, \omega) \in (S, \Omega)$ *to* $\mathbb{R}$*, find*

$$\begin{cases} \text{“}min\text{”}_{x \in S}\ G(x, \omega), \\ \textit{subject to}\ H_i(x, \omega) \leq 0,\ i = 1, 2, \ldots, m. \end{cases} \tag{1}$$

Note, however, that according to the above definition, a SCOP is not well defined, since the meaning of “min” as well as of the constraints are not clear at all [96]. In fact, how could one take a decision on x before knowing the value of ω, and how could one verify if $H_i(x, \omega) \leq 0$, if ω is not yet known? Moreover, since ω is a random variable, also $G(x, \omega)$ and $H_i(x, \omega)$ are random variables. For these reasons, the definition of SCOPs must be refined. There are several possibilities to do this, giving rise to different SCOP variants, both static and dynamic. These are also called *deterministic equivalents* of Definition 1. Let us first focus on static SCOPs, and later on dynamic SCOPs.

3.1 Static SCOPs

Definition 2 (Stochastic Integer Program - SIP)
Given a probability space (Ω, Σ, P)*, a finite set* S *of feasible solutions* x*, a real valued cost function* $G(x, \omega)$ *of the two variables* $x \in S$ *and* $\omega \in \Omega$*, and denoting by* $\mathbb{E}_P(G(x, \omega))$ *the expected value of* $G(x, \omega)$ *over* Ω *according to* P*, find*

$$\min_{x \in S} \big\{ g(x) := \mathbb{E}_P(G(x, \omega)) \big\}. \tag{2}$$

The above definition is maybe the simplest SCOP formulation, and it does not consider random constraints (observe, though, that deterministic constraints could be implicitly included in the definition of the domain S of decision variables).

In some cases the cost function G is deterministic, that is, G only depends on x and not on the random variable ω, but constraints do depend on the random variable ω. In such situation it might be impossible to enforce $H_i(x, \omega) \leq 0$ for all $\omega \in \Omega$. Thus, one could relax the notion of constraint satisfaction by allowing constraint violation, and by imposing that constraints are satisfied at least with some given probabilities. This leads to the following

Definition 3 (Chance Constrained Integer Program - CCIP)
Given a probability space (Ω, Σ, P)*, a finite set* S *of feasible solutions* x*, a real valued cost function* $G(x)$*, a set of real valued constraint functions* $H_i(x, \omega)$*, and a set of constraint violation probabilities* α_i*, with* $0 \leq \alpha_i \leq 1$ *and* $i = 1, 2, \ldots, m$*, find*

$$\begin{cases} \min\limits_{x \in S}\ G(x), \\ \textit{subject to}\ \mathrm{Prob}\{H_i(x, \omega) \leq 0\} \geq 1 - \alpha_i,\ i = 1, 2, \ldots, m. \end{cases} \tag{3}$$

Both the Stochastic and Chance Constrained Program formulations have been originally proposed in the context of Mathematical Programming applied

to SCOPs, and this field is also called in the literature Stochastic Integer Programming (SIP), a subset of the broader field of Stochastic Programming [28]. The Stochastic Programming community has a very active website [129] where updated bibliographic references and papers are available. Recent surveys on SIP include [83] and [97] (the latter overviews SIP applications in the context of location routing problems). Let us now focus on some dynamic SCOP deterministic equivalents of Definition 1.

3.2 Dynamic SCOPs

Informally, a stochastic dynamic problem is a problem where decisions are taken at discrete times $t = 1, \ldots, T$, the horizon T being finite or infinite. Decisions taken at time t may influence the random events that happen in the environment after t. In dynamic SCOPs the concept of solution used in static SCOPs is no longer valid. For example, in the dynamic TSP that we described in section 2, a tour among the set of customers known at the beginning of the day cannot be traveled as it is in practice, but it must be modified when new observations (new customers) are known. What the decision maker can do before the observation-decision process starts is to decide which *policy* (or *strategy*) to adopt, that is, to specify a set of rules that say what action will be taken for each possible random future event. For example, in the dynamic TSP, a possible policy consists in re-optimizing the portion of route among the not-yet-visited customers each time that a new customer appears. Another policy, which is less computationally expensive, but that possibly leads to a more costly tour, is to re-optimize at stages, only after a certain number of new customers has appeared. Note that in solving dynamic SCOPs one has to make a double effort: first, decide which policy to adopt, second, given the policy, solve the optimization sub-problems emerging dynamically. Both parts have influence on the final solution cost, but often the choice of the policy is due to factors that are outside the control of the decision maker. For instance, in the dynamic TSP one could be forced not to optimize every time a new customer arrives, in case customers want to know in advance the vehicle arrival time.

Among the dynamic formulations the most common ones are those belonging to the class of Stochastic Programming with Recourse (Two-stage and Multiple-stage Integer Stochastic Programs) and Markov Decision Processes.

Definition 4 (Two-stage Stochastic Integer Program - TSIP)
Given a probability space (Ω, Σ, P), a finite set S_1 of first-stage decisions x_1, a finite set S_2 of second-stage decisions x_2, and real valued cost functions f_1 and f_2, find

$$\min_{x_1 \in S_1} \big\{ g_1(x_1) := f_1(x_1) + \mathbb{E}_P\big(G(x_1, \omega)\big) \big\}, \tag{4}$$

where

$$G(x_1, \omega) := \min_{x_2 \in S_2(x_1, \omega)} f_2(x_1, x_2, \omega). \tag{5}$$

Given the above definition, solving a Two-stage Stochastic Integer Program consists in solving two problems: a DCOP for the second-stage (equation (5)), and a Stochastic Integer Program (Definition 2) for the first-stage (equation (4)). The meaning of the two-stage decision process is the following. The first-stage decision x_1 must be taken before knowing the actual value of the random variable ω. After the value of ω is observed, it may be convenient or necessary to take some other decision (the second-stage decision x_2) in order to better adapt to the new situation discovered. The second-stage decision is also called *recourse* action, because in some practical situations it has the effect of 'repairing' the consequences of an action (x_1) taken before knowing the value of the random variable. Informally, a Two-stage Stochastic Integer Program cosists in finding the best decision now, with the hypothesis that I will also take the best decision when I will know the value of the random quantities. A practical example of a Two-stage Integer Program is the Vehicle Routing Problem with Stochastic Demands (VRSPD), where a vehicle tour among a set of customers is decided, prior of knowing the actual demand of each customer. The vehicle travels along the tour, and the driver discovers the actual demand of a customer only when arriving at that customer. When a customer demand is known and the customer has been serviced, the next best decision may be to go back to the depot for replenishment, or to proceed to the next planned customer. The choice between these options is part of the second-stage optimization problem. In this context, the tour planned a priori may be interpreted as the first-stage decision x_1, while the set of return trips to the depot may be interpreted as the second-stage decision x_2.

The Two-stage Stochastic Integer Program may be easily extended to the general Multi-stage case.

Definition 5 (Multi-stage Stochastic Integer Program - MSIP) *Consider T decision stages $t = 1, 2, \ldots, T$, and correspondingly, T decision variables $x_t \in S_t$ (with S_t finite subsets depending on $(x_1, \ldots, x_{t-1}, \omega_1, \ldots \omega_{t-1})$), and T random variables ω_t belonging to probability spaces $(\Omega_t, \Sigma_t, P_t)$. The problem consists in finding*

$$\min_{x_1 \in S_1} \left\{ g(x) := f_1(x_1) + \mathbb{E}_{P_1}(G_1(x_1, \omega_1)) \right\} \tag{6}$$

where, for $t = 1, 2, \ldots, T-2$,

$$\begin{aligned} G_t(x_1, \ldots, x_t, \omega_1, \ldots \omega_t) = \min_{x_{t+1} \in S_{t+1}} \big[& f_{t+1}(x_1, \ldots, x_{t+1}, \omega_1, \ldots, \omega_{t+1}) \\ & + \mathbb{E}_{P_{t+1}}(G_{t+1}(x_1, \ldots, x_{t+1}, \omega_1, \ldots, \omega_{t+1})) \big], \end{aligned} \tag{7}$$

and

$$G_{T-1}(x_1, \ldots, x_{T-1}, \omega_1, \ldots, \omega_{T-1}) = \min_{x_T \in S_T} f_T(x_1, \ldots, x_T, \omega_1, \ldots, \omega_T). \tag{8}$$

Observe that, from the above definition, solving a Multi-stage Stochastic Integer Program consists in solving one DCOP for the last stage (equation (8)), and

$T-1$ Stochastic Integer Programs for the intermediate stages (equations (6) and (7)).

Let us now introduce the framework of Markov Decision Processes (MDP). The central concept in MDP is the *state*, which at each time step describes the knowledge about the problem (called here system). In MDP the Markov property is assumed, that is, future behavior of the system does only depend on past history through the current state and the current decision taken. Obviously, this also depends on the way a state is defined. Often, the state corresponds to the physical description of the system, but this is not always the case. We now briefly provide a standard formulation of MDP. For a complete discussion, see for instance [115].

Definition 6 (Markov Decision Process - MDP)
Consider a 4-tuple (X, A, P, C), where X is a finite state space, and A is a finite set of possible actions or decisions. At state x, we denote the set of admissible actions by $A(x)$. For each $x \in X$, $A(x)$ is a finite set. P is a state transition function, that describes the stochastic behavior of the system over time: at time t, if the current state is $x \in X$, and action $a \in A$ is chosen, then the next state is y with probability $P(y|x, a)$. P is thus a function from $X \times A(X) = \{(x, A(x)|x \in X)\}$ to a probability distribution over X. C specifies the costs of actions depending on the state in which they are performed. Taking an action a at state x has a cost $c(x, a)$. C is a function from $X \times A(X)$ to $\mathbb{R}$.

A policy π is defined as a mapping from X to $A(X)$ that associates to every x a feasible action $a \in A(x)$, and Π is the set of all possible policies. Informally, a policy tells what actions need to be taken at which state, and this is what the decision maker needs to know in order to take a decision at each discrete decision time, once the actual state of the system is known.

Let us now define the cost associated to a policy. Let $X_t, t = 0, 1, 2, \ldots, T$ be a random variable that denotes the state at time t. For a given policy $\pi \in \Pi$, and a given initial state $X_0 = x_0$, if the decision maker follows the policy π over time, a particular system path is given as a sequence of state and action $(X_0 = x_0, a_0, X_1 = x_1, a_1, \ldots, X_t = x_t, a_t, X_{t+1} = x_{t+1}, a_{t+1}, \ldots, a_{T-1}, X_T = x_T)$, where a_t is the action taken at time t and $a_t = \pi(x_t)$ with $x_t \in X$. Over the system path, the system accumulates the discounted costs defined, for $T < \infty$, as

$$\sum_{t=0}^{T} \gamma^t C(x_t, \pi(x_t)), \text{ with } \gamma \in (0, 1]. \tag{9}$$

For $T = \infty$, γ (that is called discount factor) must be smaller than 1. Given a policy π, the accumulated discounted cost (equation (9)) is a random quantity, due to the randomness of the system path. The expected discounted cost of a policy over all possible system paths may be computed as follows:

$$J(\pi) = \sum_{x_0, \ldots, x_T} P_\pi(x_0, x_1, \ldots, x_T) \left[\sum_{t=0}^{T} \gamma^t C(x_t, \pi(x_t)) \right], \tag{10}$$

with $\gamma \in (0, 1]$*, and where the probability of a particular path is*

$$P_\pi(x_0, x_1, \ldots, x_T) = \delta(x_0)\Pi_{t=0}^{T} P(x_{t+1}|x_t, \pi(x_t)), \tag{11}$$

$\delta(x_0)$ *being the probability that state* $X_0 = x_0$*.*

Given a 4-tuple (X,A,P,C) and the associated finite set of policies Π*, an MDP consists in finding*

$$\min_{\pi \in \Pi} J(\pi). \tag{12}$$

The model we have defined is known both as Markov Decision Process and Stochastic Dynamic Programming. The first term comes from the fact that for any fixed policy, the resulting model is a Markov chain. The second term comes from the family of solution methods based on Dynamic Programming [14], originally designed for solving optimal control problems. Dynamic Programming and MDP are also related to (and could be seen as part of) Reinforcement Learning [131] and Neuro-Dynamic Programming [17], which mainly focus on methods for approximating optimal solutions of MDP and for dealing with incomplete information about the state of the system. Stochastic Dynamic Programming is also related to Stochastic (Mathematical) Programming, and in particular to Multi-stage Integer Programs. For a clear exposition on the relation between these two domains, see for instance Kall and Wallace [96]).

Finding the optimal policy can be a prohibitive task unless the state and/or action space is very small, which is usually not the case. For example Value Iteration, a well known exact method for solving MDP [115], has a computational complexity polynomial in $|X|$, $|A|$, and $1/(1-\gamma)$, and one single iteration takes $O(|X|^2 \cdot |A|)$. There are several approximation algorithms (that is, algorithms that do not guarantee to find the optimal solution) that try to find good solutions in a feasible computation time via techniques such as structural analysis, aggregation, sampling feature extraction and so on. See, for example, [115, 131, 17]. Recently, also some metaheuristics have been applied to approximately solve MDP, and they are discussed in the remainder of this paper.

3.3 Objective function computation

As we have seen, all the above SCOP formulations involve the computation of one or more expected values for evaluating the objective function. As a consequence, three different situations may arise when computing SCOP objective functions:

1. closed-form expressions for the expected values are available, and the objective function is *computed exactly* based on these objective values;

2. as in case 1, closed-form expressions for the expected values are available, but the objective function is considered to be too time consuming to be always computed during optimization. Therefore, *ad-hoc and fast approximations* of the objective are designed and used during optimization (possibly alternating exact and approximated evaluations);

	Exact	Ad-hoc approximation	Simulation approximation
SIP	[53], [134]	[24], [25], [33], [34], [21], [22], [55]	[74], [75], [27], [139], [142], [143], [92], [63], [77], [78], [120], [58], [6], [5], [89], [4], [38], [80], [121], [104], [56], [45], [44], [76], [79], [114]
CCIP		[13]	[8]
TSIP	[105]	[66]	
MSIP		[119], [103], [18]	
MDP	[42], [43]	[101], [102]	

Table 2: Classification of metaheuristics papers according to SCOP formulation (rows) and type of objective function computation (columns).

3. the problem is so complex in terms of decision variables and/or in terms of probabilistic dependences, that no closed-form expression exists for the expected values, therefore, the objective function is *estimated by simulation*.

All the three above situations have been addressed by the metaheuristics literature under consideration in this survey, as summarized by Table 2. Let us now give some introductory information on the use of ad-hoc and sampling approximations in SCOPs.

3.3.1 Ad-hoc approximations

The design of ad-hoc approximations is strongly problem dependent, and no general rule exists for finding efficient approximations of the objective function. Examples of ad-hoc approximations in the literature include: the use of the objective function of a DCOP similar in some respects to the SCOP considered; the use of truncated expressions for the expected values, by neglecting terms that are estimated to be small; the use of scenarios, instead of considering the true probabilistic model. Ad-hoc approximations, if on one side accelerate the evaluation and comparison among solutions, on the other side introduce a systematic error in the computation of objective values. Usually, the systematic error cannot be reduced unless a different, more precise ad-hoc approximation is designed, and it can only be evaluated by comparison with the exact objective value. Thus, metaheuristics typically alternate exact and approximated evaluations during the optimization process. More details about the way ad-hoc approximations are used by metaheuristics will follow in section 4.

3.3.2 Simulation approximation

When a closed-form expression for the expected value(s) is not available, one common choice is to estimate expectations by Monte Carlo-type simulations. For example, in the case of the Stochastic Integer Program (Definition 2), the objective function $g(x)$ is typically approximated by the sample average

$$g_N(x) := \frac{1}{N} \sum_{j=1}^{N} G(x, \omega_j) \tag{13}$$

where $\omega_1, \omega_2, \ldots, \omega_N$ is a random sample of N independent, identically distributed (i.i.d.) realizations of the random vector ω. The sample average is also referred to as sample estimate, and the random realizations as random scenarios. In this paper, we will use these terms interchangeably.

The main difference between SCOPs requiring simulation for estimating the objective function and DCOPs, or SCOPs with exactly computable objective function is that, in the first-mentioned case, it is not possible to decide with certainty whether a solution is better than another one. This can only be tested by statistical sampling, obtaining a correct comparison result only with a certain probability. Thus, the way simulation approximation is used in metaheuristics largely depends on the way solutions are compared and the best solution among a set of other solutions is selected ('selection-of-the-best' method).

A huge research area devoted to solving problems with simulated objective function is Simulation Optimization. Following the definition given by Fu [60], Simulation Optimization means "searching for the settings of controllable decision variables that yield the maximum or minimum expected performance of a stochastic system that is presented by a simulation model." A compact picture of the field is given by the reviews of the Winter Simulation Conference [7, 111]. The latest one by Ólafsson and Kim [111] emphasizes discrete problems and practical approaches, including some references to metaheuristics. Until a few years ago, the literature on Simulation Optimization was especially focussed on theoretical results of convergence of mathematically elegant algorithms. Interestingly, as noted by Fu [59], the many new commercial software packages for simulation do not take advantage of the theoretical results of the literature. On the contrary, most of them rely on metaheuristics such as Genetic Algorithms and Neural Networks, that are more easily adaptable to complex real-world simulations, but often their integration into commercial packages lacks rigor and is not provably convergent. Fu speculates that an interesting direction of research would be the development of algorithms that take advantage of the theoretical results of the literature, but are still flexible and applicable to real-world situations, so to fill the gap between theory and practice. Indeed, recent developments in Simulation Optimization, especially relying on metaheuristics, go in this direction.

4 Metaheuristics for SCOPs

A metaheuristic is a set of concepts that can be used to define heuristic methods that can be applied to a wide range of different problems. In other words, a metaheuristic can be seen as a general algorithmic framework which can be applied to different optimization problems with relatively few modifications to make them adapted to a specific problem.

Metaheuristics that have been applied to SCOPs include: ACO, EC, SA, and TS (acronyms are explained in Table 3). For a review and comparison among these and other metaheuristics in the context of DCOPs, see for instance the paper by Blum and Roli [30], and the publications pointed to by [51]. Besides these metaheuristics, there are some algorithms such as SPM, PH, and RO (see Table 3) that could be called metaheuristics, even if they are not commonly known as such, or that make use of metaheuristics as part of the algorithmic procedure.

The following subsections focus on the cited metaheuristics. The format of subsections 4.1, 4.2, 4.3, 4.4, 4.5 describing respectively ACO, EC, SA, TS and SPM is the same: first some background information on the principles of the metaheuristic is given, then the results and issues of applying the metaheuristic to SCOPs are reviewed. For each metaheuristic, the reviewed papers have been grouped in different paragraphs, respectively focussing on:

- SCOPs with exactly computed objective or ad-hoc approximations;
- SCOPs with simulation approximation;
- Markov Decision Processes.

MDPs have been treated separately because they require a particular modeling effort for each metaheuristic, which is quite different from the modeling of static SCOPs and of TSIPs and MSIPs. Subsection 4.6 briefly gives references to the SCOP literature involving metaheuristics that are still at their early stage in the SCOP domain (PH and RO).

4.1 Ant Colony Optimization

The first algorithms based on the ant colony analogy appeared at the beginning of the nineties in a paper by Dorigo et al. [48] later published as [49]. ACO is now a widely studied metaheuristic for combinatorial optimization problems, as the recent book by Dorigo and Stützle [50] testifies.

The inspiring concept that links optimization with biological ants is based on the observation of their foraging behavior: when walking on routes from the nest to a source of food, ants seem to find not simply a random route, but a quite 'good' one, in terms of shortness, or, equivalently, in terms of time of travel; thus, their behavior allows them to solve an optimization problem. This kind of success of biological ants is entirely explained by their type of communication

Acronym	Metaheuristic
ACO	Ant Colony Optimization
EC = (EP + ES + GA)	Evolutionary Computation =(Evolutionary Programming + Evolutionary Strategies + Genetic Algorithms)
SA	Simulated Annealing
TS	Tabu Search
SMP = (BS + SBB + NP)	Stochastic Partitioning Methods =(Beam Search + Stochastic Branch and Bound + Nested Partitions)
PH	Progressive Hedging
RO	Rollout Algorithms

Table 3: Acronyms used for the metaheuristics described in this paper.

and by their way of deciding where to go: While walking, ants deposit a chemical called *pheromone* on the ground, and they tend to choose routes marked by strong pheromone concentrations. Given two initially unexplored routes, a short and a long one, between the nest and the source of food, ants choose at first randomly which one to walk. Ants that have chosen, at first by chance, the shorter route are the first to reach the food and to start their return to the nest. Therefore, pheromone starts to accumulate faster on the shorter route than on the longer one. Subsequent ants tend to follow the shorter route because it has more pheromone, thus reinforcing it more and more, and further attracting other ants on the good route.

Combinatorial problems addressed by ACO are usually encoded by a *construction graph* $G = (V, A)$, a completely connected graph whose nodes V are components of solutions, and arcs A are connections between components. Finding a solution means constructing a feasible walk in G. For example, in the TSP nodes correspond to customers, arcs correspond to streets connecting customers, and a feasible solution is a Hamiltonian path on the graph. The construction graph encoding is also used in current ACO applications to SCOPs and to dynamic optimization problems. Some examples are also described in [50].

The ACO algorithm is essentially the interplay of three procedures [46]: ConstructAntsSolutions, UpdatePheromones, and DeamonActions, as represented by Algorithm 1.

ConstructAntsSolutions is the process by which artificial ants construct walks on the construction graph incrementally and stochastically. For a given ant, the probability p_{kl} to go from a node k to a feasible successor node l is an increasing function of τ_{kl} and $\eta_{kl}(u)$, where τ_{kl} is the pheromone on arc (k, l), and $\eta_{kl}(u)$ is the *heuristic* value of arc (k, l), which should be a reasonable guess of how good arc (k, l) is (for example, in the TSP η_{kl} is the reciprocal of the distance between customer k and customer l). The heuristic value may depend on the partial walk u.

UpdatePheromones is the process by which pheromone is modified on arcs. Pheromone may be both increased and decreased. Pheromone is modified (decreased) by each ant on each arc as soon as it is added to a partial walk on the construction graph, this operation is called *local update*. Moreover, pheromone is further modified (increased) on selected good solutions to more strongly bias the search in future iterations, and this operation is called *global update*. Decreasing pheromone on selected arcs is important, in order to avoid too rapid convergence of the algorithm to suboptimal solutions. Interestingly, pheromone decreases also in the biological environment, due to evaporation.

DeamonActions are centralized operations, such as comparing solution values among ants in order to find the best solution, or running a local search procedure.

Algorithm 1 Ant Colony Optimization (ACO)

```
while termination condition not met do
  ScheduleActivities
    ConstructAntsSolutions
    UpdatePheromone
    DeamonActions
  end ScheduleActivities
end while
```

4.1.1 ACO for SCOPs

The investigation of ACO applications to SCOPs is at its early stages, the first works being appeared at conferences after year 2000. Nevertheless, the ACO literature contains both theoretical and experimental works that cover both static and dynamic SCOPs.

Exact objective or ad-hoc approximation The first SCOPs that have been addressed by ACO are the probabilistic TSP (PTSP), in Bianchi et al. [24, 25] and Branke and Guntsch [33, 34], and the vehicle routing with stochastic demands (VRPSD) in Bianchi et al. [21, 22]. These problems are Stochastic Integer Programs with closed form expression for the objective function, that is, the objective function is computable a priori, independently from particular random realizations of the stochastic variables.

The PTSP and the VRPSD have in common the fact that their solution structure (and the corresponding construction graph) is very similar to their deterministic counterpart (the TSP, respectively capacitated VRP). The main difference with the respective deterministic counterpart problem is the much higher computational complexity of the objective function in the stochastic version of the problem. In the PTSP, the objective function is computable in $O(n^2)$ time, n being the number of customers, while in the TSP it only requires $O(n)$ time. In the VRPSD, the objective requires $O(nKQ)$, where n is the number of customers, K is the number of possible demand values of each customer,

and Q is the vehicle capacity, while the capacitated VRP objective only requires $O(n)$ time. The fact that the difference between the stochastic and deterministic versions of these problems mainly lies in the objective function makes them particularly appropriate for studying a first application of ACO to SCOPs. In fact, in this case it is possible to apply to the stochastic problem an ACO algorithm originally designed for the deterministic problem with almost no modifications.

In [24, 25], the authors experimentally investigate on the PTSP two versions of ACO: ACS and pACS. ACS, that was originally designed for the TSP by Gambardella and Dorigo [61] and by Dorigo and Gambardella [47], solves the PTSP using the objective function of the TSP (the length of a Hamiltonian path) as a rough but fast approximation of the PTSP objective function. The second version of ACO considered in [24, 25], pACS, is identical to ACS except from the fact that it uses the PTSP objective function (the expected length). Such difference implies that pACS considers as good solutions different solutions with respect to ACS, and so pACS reinforces pheromone (during global updating) on different solutions with respect to ACS, with the consequence that ants in pACS converge on different solutions. Note, however, that ACS and pACS use the same, TSP specific, heuristic information (the reciprocal of the distance between two customers). Experimental results on PTSP instances with homogeneous customers probabilities have shown that pACS is better than ACS, except for the case when the customers probabilities are close to 1, in which case ACS is more efficient than pACS. This means that the overhead of the time consuming PTSP objective function is not justified in those cases where the approximate objective function, which can be computed much faster, is close enough to the exact one. The idea to employ faster approximations of the exact objective function has been further developed in [33, 34]. The authors propose an ad-hoc approximation of the expected cost that neglects the least probable customers configurations. This approximation is shown experimentally to accelerate convergence without significantly worsening the solution quality. Another issue addressed by [33, 34] is the design of PTSP-specific heuristics to guide the ants construction process. The authors experimentally analyze different heuristics, and show that one of them indeed improves the quality of solution constructed by ants, but at the cost of a higher computational time.

An important aspect in designing ACO for SCOPs (but also for classical DCOPs), is the application of a local search procedure to improve solutions found by ants (the local search is part of the DeamonActions of Algorithm 1). In order to be competitive with state-of-the-art algorithms, it has been *necessary* for ACO algorithms to use a local search both in the PTSP [34] and the VRPSD [21]. Unfortunately, designing an effective local search for a stochastic problem with a computationally expensive objective function may be a quite challenging task. The reason is that in local search it is very important to compute efficiently the change, or 'delta', of the objective function between two neighboring solutions. When the objective function is complex like in most SCOPs, it is difficult to find a delta expression which is both exact and fast to be computed. For the PTSP it has been possible to derive for two local search operators, the 1-shift and the 2-p-opt, recursive, fast and exact expressions for

the objective function delta [26, 23]. The 1-shift and 2-p-opt are very efficient, since they can explore the whole neighborhood of a solution in $O(n^2)$ time, the same time it would take for the same operators in the TSP. For the VRPSD, a local search operator is available, the OrOpt, with an efficient delta expression which is not exact, but approximated, and has been introduced by Yang et al. in [140] (we will call this 'Yang approximation'). In Bianchi et al. [21, 22], besides the Yang approximation, one based on computing the length difference between two neighboring solutions has been considered. This last approximation is equivalent to treat a VRPSD solution (which is a Hamiltonian path) like a solution for the TSP, and it is faster but less accurate than the Yang approximation. In [21, 22], the impact of using the two above types of delta approximation has been tested on several metaheuristics, namely ACO, EC, SA, TS, and Iterated Local Search. In ACO, the use of the rough but efficient TSP approximation lead to better results than the Yang approximation (even though ACO was not able to reach the quality of the best performing metaheuristics, that were Iterated Local Search and EC).

Sampling approximation When ACO is applied to this type of problems, the DeamonActions procedure (Algorithm 1) must implement ways of performing statistical tests for comparing the sample average values of the solutions generated by ants, in order to select the best solution (or a set of best solutions). Sampling could also be used in ConstructAntsSolutions, in order to estimate heuristic values $\eta_{k,l}(u)$, when the chosen heuristic depends on random variables.

The first sampling-based ACO, called S-ACO, has been proposed and analyzed by Gutjahr in two papers [74, 75]. The first paper [74] theoretically analyzes S-ACO, by proving convergence to the optimal solution with probability one. The second paper [75] experimentally studies S-ACO on two stochastic routing problems, the PTSP, and the TSP with time windows and stochastic service times (TSPTW). S-ACO has been applied in a third paper by Rauner et al. [116] to a policy optimization problem in healthcare management. Algorithm 2 summarizes the functioning of S-ACO, showing in particular how sampling is used; for details about procedures ConstructAntsSolutions and UpdatePheromone, see [74, 75]. In every iteration, after ants have constructed their solutions x_σ, only one ant solution x is selected for being further compared with the current best solution (step 3 of Algorithm 2). Interestingly, for the sake of convergence, it does not matter how the ant solution is selected [74]. A possible way to do it, which has been chosen in [75], is to evaluate each x_σ on a same random scenario drawn specifically for a given iteration, and to take x as the solution with the best value. In the case of the more complex problem treated in [116], selecting x_σ based on a single random scenario turned out as suboptimal; better results were obtained by choosing several (but not too many) scenarios. After x has been selected, it is then again evaluated, together with the current best solution x^*, in order to decide whether it is better than x^*. This is done by estimating x by sampling over N_k scenarios ω_ν and x^* over N_k scenarios ω'_ν. In

Algorithm 2 S-ACO

1: **for** iteration $k = 1, 2, \ldots$ **do**
2: ConstructAntsSolutions [s ants construct their walk x_σ, $\sigma = 1, 2, \ldots, s$ on the graph G]
3: from $\{x_1, \ldots, x_s\}$ select a walk x;
4: **if** $k = 1$ **then**
5: set $x^* = x$ [x^* is the current approximation of the optimal solution]
6: **else**
7: based on N_k independent random scenarios ω_ν, compute a sample estimate $g_k(x) = 1/N_k \sum_{\nu=1}^{N_k} G(x, \omega_\nu)$ of x;
8: based on N_k independent random scenarios ω'_ν, compute a sample estimate $g_k(x^*) = 1/N_k \sum_{\nu=1}^{N_k} G(x^*, \omega'_\nu)$ of x^*;
9: **if** $g_k(x) < g_k(x^*)$ **then**
10: set $x^* = x$;
11: **end if**
12: **end if**
13: UpdatePheromone
14: **end for**

the convergence proof of [74], it has been necessary to impose that ω_ν and ω'_ν are independent, but in practice [75], if $\omega_\nu = \omega'_\nu$ S-ACO also performs well. The number of sample scenarios N_k is a critical parameter of S-ACO: if too small, the estimate and comparison of solutions will be often faulty, but if N_k is too big, the computational time required for one solution evaluation could become a problem. As shown in [74], for proving convergence it is sufficient that N_k increases linearly with the iteration number k. This result is interesting especially if compared with the faster than quadratic increase recognized as necessary for the corresponding SA approach in [77, 89]. In practical implementations of S-ACO, it may be more convenient to choose the sample size N_k adaptively, based on some statistical test. In [75], one version of S-ACO establishes the sample size by means of a parametric statistical test: N_k is gradually increased till when the difference between the sample estimation for the two solutions being compared is larger than 3 times their estimated standard deviation. This kind of sample schedule, also known as variable-sample strategy, has been theoretically analyzed in the context of random search algorithms by Homem-de-Mello [90].

More recently, the ACO/F-Race algorithm has been proposed by Birattari et al. [27], where at each iteration the selection of the new best solution (steps 3 to 12 of Algorithm 2) is done with a procedure called F-Race, which is more sophisticated than the simple parametric test of S-ACO. As explained in [27], F-Race consists in a series of steps at each of which a new scenario ω is sampled and is used for evaluating the solutions that are still in the race (at the beginning, all solutions generated by ants in a given iteration, together with the current best solution, are in the race). At each step, a Friedman test is performed and solutions that are statistically dominated by at least another one are discarded

from the race. The solution that wins the race is stored as the new current best solution. Preliminary experiments on homogeneous instances of the PTSP problem have shown that ACO/F-Race improves over the parametric procedure adopted by S-ACO.

Markov Decision Processes To our knowledge, there are only two papers that use ACO to solve MDP, the first one by Chang et al. [42] and the second one by Chang [41]. Both papers design ACO algorithms to solve MDP, and theoretically analyze their properties by providing convergence proofs.

Chang et al. [42] focus on MDP with infinite horizon (that is, $T = \infty$). This simplifies a little the problem, because in this case the optimal policy is stationary, that is, it does not depend on time. The construction graph on which ACO algorithms proposed in [42] are based is represented in Figure 2. The states in X

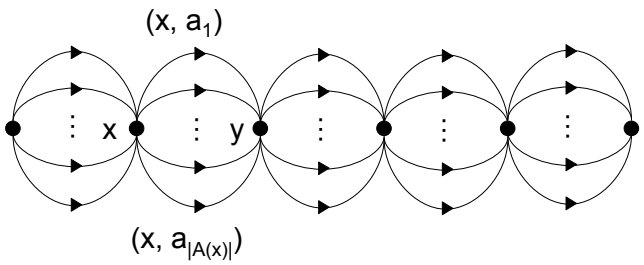

Figure 2: Construction graph of the ACO algorithm proposed in [42] for solving Markov Decision Processes with stationary policies.

are arranged in an arbitrary order $\mathcal{O}$; each state $x \in X$ corresponds to a vertex of the construction graph, and each arc of the construction graph corresponds to a pair (x, a) of state $x \in X$ and an admissible action $a \in A(x)$. The direction of the arc goes from the current state x to the next state with respect to the order $\mathcal{O}$. A particular ant traverses all the states in the construction graph following the order $\mathcal{O}$ from the first state of $\mathcal{O}$. When it moves from a state x to a state y, it traverses randomly the arc (x, a) with a transition probability that depends on the pheromone on that arc, and on some heuristic appropriately defined (see [42] for details). Once an ant finishes the tour, it has generated a stationary policy, which, in the context of MDP, is a candidate solution to the problem. In [42], two ACO versions are proposed, ANT-PI and ANT-TD. The first one is inspired by a well-known exact method for MDP, policy iteration (PI) [115], and assumes that the state transition probabilities P of the MDP system are known. The second one considers the case of unknown P, and uses one the well-known reinforcement learning algorithms called temporal difference learning (TD) [131] for evaluating the average value of a given policy. It is proven that both ANT-PI and ANT-TD probabilistically converge to the optimal solution. For practical implementations, due to the high computational complexity

inherent to the problem, the authors suggest that parallel implementations of the proposed ACO algorithms are used.

In Chang [41], the focus is on MDP with completely unknown system dynamics, that is, unknown state transition probability P *and* unknown costs C. A finite horizon T is considered and, for simplicity, it is assumed that every action is admissible at every state. An ACO algorithm called ANT-EE is proposed, which is based on a TD algorithm for evaluating a policy, similarly to the ANT-TD algorithm of [42]. Theorems are provided that show that ANT-EE has the same convergence properties as Q-learning [136], one of the most important basic techniques in reinforcement learning.

4.2 Evolutionary Computation

EC [20] is a collective term for all variants of optimization algorithms that are inspired by Darwinian evolution. In this context, a solution to a given optimization problem is called *individual*, and a set of solutions is called *population*. The basic structure of an EC algorithm is represented by Algorithm 3. Every iteration of the algorithm corresponds to a *generation*, where certain operators are applied to some individuals of the current population to generate the individuals of the population of the next generation. Typical operators are those of *recombination*, that recombine two or more individuals to produce new individuals, and *mutation*, that modify single individuals to obtain self-adaptation. At each generation, only some individuals are selected for being elaborated by recombination and mutation operators, or for being just repeated in the next generation without any change, on the base of their fitness measure (this can be the objective function value, or some other kind of quality measure). Individuals with higher fitness have a higher probability to be selected.

In the literature there are mainly three different categories of EC that have been developed independently from each other: Evolutionary Programming (EP), proposed by Fogel et al. in 1966 [57], Evolutionary Strategies (ES) proposed by Rechenberg in 1973 [117], and Genetic Algorithms proposed by Holland in 1975 [88]. Presently, algorithms that fall in the EP and ES category mostly apply to continuous optimization problems, while GA are more specific for discrete and combinatorial optimization problems. Recent overviews about EC include Hertz and Kobler [86], and Calégari et al. [39]. For the convergence properties of EC and GA, see for instance Rudolph [123], Vose [138], and Reeves and Rowe [118].

4.2.1 EC for SCOPs

There is a very large amount of literature on applying EC to optimization problems 'under uncertainty', such as problems with noisy fitness, with time varying and dynamic fitness, and with approximated fitness. For a recent survey on how the EC literature addresses different types of uncertainty, see Jin and Branke [94]. Other reviews include a book by Arnold [9], and a paper by Beyer [19]. Although SCOPs are a particular case of optimization under uncertainty, the

Algorithm 3 Evolutionary Computation (EC)

```
P = GenerateInitialPopulation()
while termination condition not met do
  P' = Recombine(P)
  P'' = Mutate(P')
  Evaluate(P'')
  P = Select(P'' ∪ P)
end while
```

translation to the SCOP domain of the results from the EC literature on optimization under uncertainty is not easy. The main difficulty is that most papers focus on continuous optimization, and they often restrict their attention to the optimization problem characterized by ad-hoc test functions, such as the 'spherical' objective function $f(\mathbf{x}) = \mathbf{x}^T\mathbf{x}$, $\mathbf{x} \in \mathbb{R}^N$. Also when discrete optimization problems are considered, experiments are often restricted to the 'onemax bit-counting' function, which can be regarded as the counterpart of the spherical objective function in binary search spaces.

In the following, we outline the contributions of EC in the SCOP domain, and we also highlight the main ideas and methods proposed for problems under uncertainty that may be relevant for SCOPs, even if they have not been directly tested on specific problems from this domain.

Exact objective or ad-hoc approximation The EC literature addressing this kind of problems may be roughly divided in two groups. In the first group ([53, 105]) EC algorithms use the exactly computable objective function as it is, even if computationally expensive, while in the second group ([21, 22], and references cited in [93]) EC exploits also computationally more efficient objective function (fitness) approximations. Let us briefly analyze these two groups of papers.

In [53] Easton and Mansour apply a distributed GA to three different labor scheduling problems, one of which is formulated as a stochastic goal programming problem. Their algorithm operates in parallel on a network of three workstations. Separate sub-populations evolve independently on each processor, but occasionally the fittest solutions migrate over the network to join the other sub-populations. Also infeasible solutions are accepted (with a fitness penalty) in order to encourage the exploration of promising regions of the search space. The proposed GA is compared experimentally to a SA and a TS metaheuristic previously developed by other authors (respectively in [36, 37] and in [54]), and it is shown to outperform both of them.

In [105] Guo and Mak consider a vehicle routing problem with stochastic demand and soft time windows, which is formulated as a Two-stage Stochastic Integer Program (Definition 4). The authors propose an EC algorithm called Age-GA where, instead of being replaced by their offspring after each iteration, individuals may grow up and generate new offspring continuously before death,

and the population comprises individuals from various age-groups. With the same amount of computational effort, it is possible to use a larger population size in Age-GA than in a canonical GA. The paper shows that, on a set of eighteen randomly generated instances, Age-GA outperforms a canonical GA without the aging mechanism.

To the group of papers using in SCOPs efficient approximations of the fitness function belong [21, 22] by Bianchi et al. (also cited in section 4.1), that compare a simple EC with other metaheuristics (ACO, SA, TS, and Iterated Local Search) for the VRPSD. Similarly to the other metaheuristics, EC is integrated with the OrOpt local search operator, where two approximations for the objective value difference between neighboring solutions have been tested, the Yang and the TSP approximation. The exact VRPSD objective function is used for accepting a new solution in the local search, and for the selection of a new population. EC, like ACO and Iterated Local Search, performs better with the TSP approximation. Interestingly, EC is improved even more when in [22], instead of OrOpt, a more TSP-specific local search (3-opt) is used. EC, together with Iterated Local Search, is shown to be the best performing among the tested metaheuristics.

Here, it is useful to note that there is a thread in the EC literature that focuses on the use of computationally efficient approximations of the original fitness in continuous optimization problems. Some aspects of this issue that are developed in the context of continuous optimization may be relevant to SCOPs as well. Fitness approximations are also known as approximate models, meta-models or surrogates. A comprehensive survey on fitness approximation in EC has been written by Jin [93]. This growing research area is particularly oriented to continuous optimization problems with extremely time consuming objective function computations, such as, for instance, structural design optimization [10], where one single fitness evaluation may take over ten hours on a high-performance computer. The issue of how the approximate model can be incorporated in the EC algorithm, which has been widely addressed by the EC literature on fitness approximation, is quite independent from the continuous or discrete nature of the optimization problem. Nevertheless, most of the ideas still haven't been applied to SCOPs. For a review and pointers to existing literature, see Section 4 of [93].

Sampling approximation The EC literature about optimization with noisy fitness function is also relevant for SCOPs with sampling estimated objective function. In fact, noise is mostly assumed to be additive with zero-mean, which is the case when Monte Carlo sampling is used to estimate the objective function. Section II of Jin and Branke [94] is a good overview about the methodological approaches used in EC to deal with noisy fitness functions. The authors identify three main strategies for dealing with noise: explicit averaging, implicit averaging, and modifying the selection process.

Explicit averaging corresponds to the computation of sample averages of the fitness function performing repeated measures of the noisy fitness and computing

their average. This is very similar to the Simulation Optimization technique we have illustrated in section 3.3.2. Aizawa and Wah [2] propose either to increase the number of samples with the generation counter, or to higher the number of samples for individuals that have a high estimated variance. Stagge [128] proposes to adjust the number of samples according to the probability of an individual of being among the μ best ones. Branke and Schmidt [35] also propose an adaptive sampling method that takes additional samples of two individuals participating in a tournament until the normalized fitness difference between the two individuals falls below some threshold. The normalized fitness difference is obtained dividing the difference of the observed fitnesses by the standard deviation of the difference: $(g_N(x) - g_N(y))/\sigma_d$. Cantú-Paz [40] uses an adaptive sampling method that consists in taking the smallest number of samples necessary to make a decision between competing individuals during the selection process. Their approach is very similar to Branke and Schmidt's, but differs in that they take samples one at a time from the individual with the highest observed variance, and they use standard statistical tests to select the winner of the tournament with a certain probability. Similar to the approach of Cantú-Paz, Teller and Andre [133] propose a method that allocates varying numbers of samples to evaluate individuals. Individuals are initially evaluated with a small number of samples, and are further evaluated only if there is some chance that the outcome of the tournaments they participate in can change. A similar technique has been developed by Giacobini et al. [67].

The second type of strategy for dealing with noise in EC is implicit averaging. Its aim is to reduce the influence of noise by using a large population size, instead of performing more fitness measures of a single individual. The intuition behind implicit averaging is the following ([94], p.305): Because promising areas of the search space are sampled repeatedly by the EC algorithm, and there are usually many similar solutions in the population, when the population is large, the influence of noise in evaluating an individual is very likely to be compensated by that of a similar individual. This can be regarded as an implicit averaging effect. In the literature, conflicting conclusions have been reported on whether it is more effective the use of implicit or explicit averaging, given a fixed fitness evaluation number per generation is allowed. For a summary about the history of implicit averaging we direct the reader to [94] and to the references cited therein.

The third strategy used in EC to reduce the influence of noise is modifying the selection process of individuals. One example is to accept an offspring individual only if its fitness is better that that of its parents by at least a predefined threshold, as done by Markon et al. [106]. Beielstein and Markon [11] study the relationship between threshold selection and hypothesis testing techniques. Another way of modifying the selection process with respect to standard EC is to eliminate random choices during selection, in order to exploit the uncertainty due to the noisy fitness as a sort of randomization effect. This method has been proposed by Branke and Schmidt [35].

Convergence properties of EC and the dynamics of the fitness function when noise is present have been analyzed in several papers, for example by Miller and

Goldberg [108] and by Beyer [19].

In all the above cited papers aiming at reducing the influence of noise via implicit and explicit averaging, or via modifications of the selection process, the computational experience is unfortunately limited to ad-hoc continuous or discrete test functions. It appears that an experimental validation of the various techniques in the SCOP domain is still missing in the literature. In fact, the few papers applying EC to SCOPs that we are going to outline below, either use very simple techniques for dealing with noise or rely on methods that are unrelated to the main EC literature on noisy fitness functions.

Watson et al. [139] address a stochastic warehouse scheduling problem where the objective function must be estimated by simulation. The authors consider a GA, a solution construction heuristic specific for that problem, two local search and a random search algorithm. Two versions of the GA and the local search algorithms are considered where the (set of) starting solution(s) is randomly generated in one case, and provided by the constructive heuristic in the other case. In order to keep the run time of the algorithms feasible, the simulator is used in a fast but inaccurate mode. Only final solutions are eventually evaluated with a more accurate - two order of magnitude slower - simulator mode. The constructive heuristic exploits specific knowledge about the internal states of the simulator in order to construct a solution. Instead, in the GA and local search algorithms the simulator is used as a black box, that, provided a solution, returns a real value indicating the solution quality. Experimental results show that GA initialized with the domain-specific construction heuristic outperforms all the other algorithms. Moreover, all algorithms perform worse when initialized by random solutions. The results also highlight an interesting phenomenon related to the use of a black box, fast but inaccurate simulator for the evaluation of solutions during the execution of the GA. As better and better solutions according to this simulator are found, it is observed that the correlation with solution values given by the slow-accurate simulator (evaluated a posteriori) decreases. This implies that the final solution returned by the GA as best solution may be quite bad with respect to the nearly-exact objective value. It is reasonable to think that this is a general phenomenon that can happen in any metaheuristic exploiting a non-exact or noisy objective function evaluation, particularly when estimating the objective function by sampling and with a fixed (low) number of samples. One possibility to overcome this problem is to keep in memory a set of promising solutions encountered during the execution of the algorithm, and to evaluate them a posteriori with the accurate simulator, or to apply more sophisticated adaptive sampling techniques. Yoshitomi [142] and Yoshitomi and Yamaguchi [143] use GA for solving the stochastic job-shop scheduling problem. In both papers, the best solution is extracted among the set of solutions that have been more frequently present through the generations of the GA. In [143], Monte Carlo sampling is used to select among the set of most frequent solutions the best final solution. Other applications of GA based on Monte Carlo sampling are the ones by Sudhir Ryan Daniel and Rajendran [130] applying GA to the inventory optimization problem in a serial supply chain, and by Jellouli and Châtelet [92] using GA for addressing a supply-chain

management problem in a stochastic environment.

Markov Decision Processes Chang et al. [43] propose an EC algorithm called Evolutionary Policy Iteration (EPI) for solving inifinite horizon discounted reward MDPs. EPI is particularly suited for problems where the state space is small but the action space is very large. This situation makes the use of the well-known policy iteration (PI) algorithm [115] for solving MDPs impractical, since PI must perform a maximization over the entire action space. EPI eliminates the need of maximizing over the entire action space by directly manipulating policies via a method called policy switching that generates an improved policy from a set of given policies. The computation time for running the policy switching is on the order of the state space size. The algorithmic structure of EPI is that of standard GA, with appropriate modifications and extensions required for the MDP setting. EPI iteratively generates a population or a set of policies such that the performance of the best policy for a population monotonically improves with respect to a defined fitness function. In [43], it is proved that EPI converges with probability one to a population whose best policy is an optimal policy.

Lin et al. [101, 102] focus on finite horizon partially observed Markov decision processes (POMDPs), an extension of MDPs that consider situations such as: State observations are costly; sensors that measure the current state value are noise-corrupted; at least part of the current state value is inaccessible. In [101, 102], a GA is developed for finding a good approximation of the value function. In [102], the GA is combined with a mixed integer programming algorithm, and this combination is capable of determining the value function and of finding the optimal policy in less computation time than other exact methods.

Yokoama and Lewis III [141], address a stochastic dynamic production cycling problem whose original formulation is that of an MDP. In order to reduce the search space dimensionality, the problem is first re-formulated in a two-level problem. The first level consists in a DCOP that, in order to be solved, needs that a series of MDP sub-problems (constituting the second level) are solved. The first level DCOP is addressed by a GA, which calls dynamic programming algorithms as sub-routines to solve the second level MDPs.

4.3 Simulated Annealing

The SA algorithm has been introduced in the area of combinatorial optimization by Kirkpatrick et al. [98]. It relies on a model developed by Metropolis et al. [107] for simulating the physical annealing process, where particles of a solid arrange themselves into a thermal equilibrium. An introduction to SA can be found in van Laarhoven and Aarts [137] or Aarts and Korst [1].

The standard type of applications concerns combinatorial optimization problems of the form

$$min_{x \in S}\ g(x),$$

where S is a finite set of feasible solutions. The algorithm uses a pre-defined

neighborhood structure on S. A control parameter which is called "temperature" in analogy to the physical annealing process governs the search behavior. In each iteration, a neighbor solution y to the current solution x is computed. If y has a better objective function value than x, the solution y is "accepted", that is, the current solution x is replaced by y. If, on the other hand, y has a worse objective function value than x, the solution y is only accepted with a certain probability depending on (i) the difference of the objective function values in x and y, and (ii) the temperature parameter.

In pseudocode, the SA algorithm can be represented as follows (cf. [1], p. 16):

Algorithm 4 Simulated Annealing (SA)

Initialize state x and temperature parameter T_1;
for iteration $k = 1, 2, \ldots$ **do**
  select y randomly from $S(x)$;
  if $g(y) \leq g(x)$ **then**
    set $x = y$;
  else if $\exp\left(\frac{g(x)-g(y)}{T_k}\right) \leq$ uniform[0,1] **then**
    set $x = y$;
  end if
  update T_k to T_{k+1};
end for

Therein,

- x and y are feasible solutions from S;
- T_1,$T_2, \ldots$ is a (usually decreasing) sequence of values for the temperature parameter; the update of the values T_k is done according to a so-called *cooling schedule*;
- the sets $S(x)$ form the pre-defined *neighborhood structure*: to each feasible solution $x \in S$, a set $S(x) \subseteq S \setminus \{x\}$ of "neighbor solutions" is assigned;
- uniform$[\alpha, \beta]$ is a procedure selecting a uniformly distributed (pseudo-)random number from the interval $[\alpha, \beta]$.

Several results showing convergence of SA to the set of optimal solutions under suitable cooling schedules have been obtained by diverse authors, for example Geman and Geman [64], Gelfand and Mitter [62], or Hajek [81]. Essentially, convergence can be assured by a cooling schedule where T_k is decreased as $\Gamma / \log k$, with sufficiently large Γ. In practice, cooling is usually done faster for computation time reasons. (For more details, see [1].)

4.3.1 SA for SCOPs

In the literature, several extensions of the SA algorithm above have been suggested for treating Stochastic Integer Programs (Definition 2), both in the case

of exact objective and ad-hoc approximation, and sampling approximation.

Exact objective or ad-hoc approximation One early application of SA in the context of SCOPs is due to Teodorović and Pavković [134]. The authors address a VRPSD with multiple vehicles, and use SA in two stages, first for partitioning the customers among the different vehicles, and second to improve the single vehicle routes. In this preliminary work, computational results are reported only for one instance of 50 customers.

More recently, the already cited papers [21, 22] by Bianchi et al. (see section 4.1 and 4.2) have applied to the VRPSD a simple SA algorithm, together with other metaheuristics (ACO, EC, TS, and Iterated Local Search). Similarly to the other metaheuristics, two approximations for the objective value difference between neighboring solutions generated according to the OrOpt scheme have been tested, the Yang and the TSP approximation. Differently from what happens for ACO, SA performs better when using the more accurate but more computationally expensive Yang approximation. On average, SA does not perform significantly different from ACO, and it is not able to reach the quality of the best performing metaheuristics, that are EC and Iterated Local Search.

Sampling approximation Algorithm 5 shows a typical basic structure of an SA modification to the solution of Stochastic Integer Programs (Definition 2) with sampling estimated objective function. The approaches from the literature outlined below follow this general scheme. Differences stay particularly in the way step 5 (estimation of the objective value), step 11 (choice of a new approximation of the optimal solution), and step 12 (temperature level) are implemented in Algorithm 5.

Algorithm 5 Stochastic Simulated Annealing (SSA)

1: Initialize state x, temperature parameter T_1 and sample size N_1;
2: Set $x^* = x$ [x^* is the current approximation of the optimal solution];
3: **for** iteration $k = 1, 2, \ldots$ **do**
4: select y randomly from $S(x)$;
5: compute sample average estimates $g_k(x)$ and $g_k(y)$ for the costs in x resp. y;
6: **if** $g_k(y) \leq g_k(x)$ **then**
7: set $x = y$;
8: **else if** $\exp\left(\frac{g_k(x)-g_k(y)}{T_k}\right) \leq$ uniform[0,1] **then**
9: set $x = y$;
10: **end if**
11: compute a new current approximation x^* of the optimal solution;
12: update T_k to T_{k+1};
13: update N_k to N_{k+1};
14: **end for**

Gelfand and Mitter [63] investigate the case where the observation of the

objective function $g(x)$ is disturbed by random noise W_k in iteration k of the SA process, such that instead of $g(x)$, the estimate $g_k(x) = g(x) + W_k$ is observed. They show that if W_k is normally distributed with mean zero and variance σ_k^2, if certain conditions on the values σ_k and on acceptance probabilities are satisfied, and if a suitable cooling schedule (ensuring convergence of ordinary SA) is used, then the convergence property of ordinary SA remains valid.

Gutjahr and Pflug [77] follow a similar approach by showing that under suitable conditions on the "peakedness" (Birnbaum [29]) of the noise distribution, convergence of the current solutions to the set of optimal solutions can be guaranteed. To be more specific, let us call a symmetric distribution μ_1 more peaked around zero than a symmetric distribution μ_2, if for all $t > 0$, the probability mass on the interval $[-t, t]$ is larger or equal under μ_1 than under μ_2. Then, if the distribution of the noise W_k is more peaked around zero than a normal distribution $N(0, \sigma_k^2)$, where $\sigma_k = O(k^{-\gamma})$ with a constant $\gamma > 1$, the distribution of the solution in iteration k converges as $k \to \infty$ to the uniform distribution on the set of global optimizers, provided that a suitable cooling schedule (ensuring convergence of ordinary SA) is used. Decreasing σ_k with the required rate can be achieved by increasing the sample size N_k more than quadratically in k, that is, by imposing that $N_k = O(k^{\mu})$ with $\mu > 2$. An application of the technique of [77] to a discrete time/cost tradeoff problem in activity planning has been reported in [78].

Other approaches have been presented by Roenko [120], who proposes to store the feasible solutions produced during the execution of the algorithm and to compare them with the solution generated in each current iteration, and by Fox and Heine [58], who derive a convergence result based on the assumption that with probability one, the objective function estimates $g_k(x)$ coincide after some finite time with the true objective function values $g(x)$, as it can be achieved by consistent estimators in the case of only finitely many *possible* objective function values. The last assumption can also be relaxed, if some more complicated condition can be verified, but Fox and Heine argue that in each computer representation, objective function values are taken from some finite domain (given by the machine number precision) anyway. The algorithm indicated by Fox and Heine does not use independent sampling from scratch in each iteration, as it is done in [63] and [77], but cumulates the sampling results, which is of course advantageous from a computation time viewpoint.

Alrefaei and Andradóttir [6] pursue a different idea by keeping the temperature parameter T_k *constant* during the process instead of decreasing it toward zero (as usual in ordinary SA). To obtain convergence, two alternative techniques are suggested. The first, let us call it A1, consists in the following procedure: In each iteration k, for the current solution x chosen in this iteration, a counter $V_k(x)$ is increased by one in order to register how often x has been visited since the start of the algorithm. The current number $V_k(x)$ of visits is divided by the number $D(x)$ of neighbors of x. The estimated optimal solution x^* in iteration k is then defined as that solution $x^* = x$ for which $V_k(x)/D(x)$ is maximal, among all the solutions x that have been encountered so far. The second technique, let us call it A2, is to estimate the objective function value of solutions x and y

(step 5 of Algorithm 5), by cumulating previous estimates of x and y (if any), and then, choose as new approximation x^* of the optimal solution at iteration k the solution with the smaller estimated objective value, among all solutions evaluated so far. Both A1 and A2 compute sample averages with an increasing number of samples at each iteration k.

Alrefaei and Andradóttir show that both alternatives guarantee, under mild conditions, convergence with probability 1 to the set of optimal solutions. Their article also reports on experimental comparisons showing a superiority of the introduced new algorithms over the previous approaches in [63], [77] and [58]; among the two new algorithms, A2 turns out to yield better results than A1. The experiments are restricted to a test instance with only 50 feasible solutions, therefore it is not clear whether the results can be generalized to larger search spaces; nevertheless, the empirical findings give some evidence that using the solution with best objective function estimate so far as the proposed solution may be a very good choice. Interestingly, for the considered test instance, a random-search-like neighborhood structure including all elements of S (different from x) into the neighborhood $S(x)$ of x produces, for all tested algorithms, better results than a more restricted neighborhood. This seems to indicate that in the stochastic case, the hill-climbing feature of SA gains importance only for larger solution spaces S.

A further important contribution of [6] is that the article discusses optimization both in a transient and in a steady-state simulation context. It is shown that if $g(x)$ is given as the expectation of a functional $G(x, \omega)$ of a stochastic process in either a transient or a steady-state situation, then the theoretical result derived for the simple static SCOP case (corresponding to our Definition 2) still remains valid.

One practical limitation of approaches such as the two just described by Alrefaei and Andradóttir [6] and the one by Roenko [120] is that they require the storage of information about all or most of the solutions encountered by the algorithm, and this is an infeasible task for problems that have a combinatorial nature.

Alkhamis et al. [5] use again a decreasing cooling schedule for the parameters T_k. They propose to decide on acceptance or rejection of a neighbor solution y by means of a *statistical significance test*: A confidence interval for the difference between the true objective function values in x resp. y is computed; depending on the position of the value zero in relation to this confidence interval, the neighbor solution is judged as equal, better or worse than the current solution x. After that, the usual acceptance rule of SA is applied. The authors are able to show that on certain conditions on sample size and cooling schedule, the classical SA convergence property is still satisfied.

Homem-de-Mello [89], [90] presents a comprehensive framework for describing and analyzing variants of SA for SCOPs. The framework enables a thorough theoretical analysis and opens a broader range of flexibility in the choice of sampling distributions. Using ergodicity theory, Homem-de-Mello proves in [89] a rather general convergence theorem for a variable-sample modification of SA. The theorem includes the result in [77] as a special case, but does not make use of

any normality assumptions related to noise distributions anymore. In [90], this approach is further generalized beyond the area of SA, although the described analytical techniques and algorithmic ideas remain applicable in a SA context, as well as in the context of other metaheuristics dealing with SCOPs with objective function estimated by sampling. In particular, the author presents the interesting idea of adaptively modifying the sample size N_k during the iterations of the algorithm, in such a way that N_k is usually only increased if the result of a t-test indicates that higher accuracy of the objective function estimates is required. To preserve the convergence property, the sample size is increased at some specific points in time regardless of the t-test.

In Alkhamis and Ahmed [4], the acceptance rule based on confidence intervals developed in [5] is modified by applying the constant-temperature schedule of [6] instead of the classical decreasing temperature schedule. As the current estimated solution, the authors take the solution with the maximum (normalized) number of visits so far. Again, a convergence result is given.

There are also some purely experimental papers involving SA and SCOPs with sampling estimated objective function. The earliest is a paper by Bulgak and Sanders [38] addressing a buffer allocation problem in the context of a complex manufacturing system. The objective function to be maximized (the efficiency of the system) is estimated by means of a discrete event simulator. Similarly to [90], an adaptive sampling procedure is used, where the number of samples is gradually increased for testing whether a candidate solution is statistically better than the current best solution.

Haddock and Mittenthal [80] investigate the feasibility of using an SA algorithm in conjunction with a simulation model to find the optimal parameter levels at which to operate a system. The authors modify Kirkpatrick et al. [98] by substituting an estimate of the expected value of the system response (the objective function) in all places requiring a deterministic objective function value.

Rosen et al. [121] propose a combined procedure, called RS team method, that improves the SA of Haddock and Mittenthal [80] by initially searching for good solutions to be then employed as starting solutions by SA. The initial search for good starting solutions is done by the use of first-order linear approximations of the model, adapting the technique of response surface methodology to the case of a discrete decision space. The RS team method is tested on a simulation model of a semi-conductor manufacturing process consisting of over 40 workstations, and it is experimentally compared with the SA algorithm of Haddock and Mittenthal [80].

Bowler et al. [32] use a stochastic SA algorithm to experimentally analyze the asymptotic behavior of (sub)optimal homogeneous PTSP solutions, in the limit of pn (customers probability times number of customers) going to infinity. The PTSP objective function is estimated by sampling, and the sampling estimation error is used instead of the annealing temperature. Temperature decrease during the execution of the SA algorithm is mimicked by an increase in the accuracy of the objective function estimation, which, in turn, is obtained by increasing the number of samples.

Finally, two papers [75] and [114] focus on different metaheuristics, but in-

volve SA in experimental comparison. The paper by Gutjahr [75] that we also cited in section 4.1 focuses on S-ACO, and reports experimental comparisons between S-ACO and the SSA algorithm of Gutjahr and Pflug [77]. Pichitlamken and Nelson [114], while focusing on a Stochastic Partitioning Method that will be described in section 4.5, use the SA algorithm of Alrefaei and Andradóttir [6] as a term of comparison in the experimental analysis of their algorithm.

Although variants of SA for SCOPs have received a great deal of attention in the last decade, such that, for example, the question under which conditions convergence to the optimal solution is ensured can now be considered as relatively well understood, there is a comparably smaller body of comprehensive *experimental* results aiming at interesting questions such as: Which properties of the problem instance make which algorithmic variant well-suited? In particular, there seems still to be little empirical knowledge about the influence of the search space size on the performance of the single variants.

4.4 Tabu Search

The main ideas characterizing the TS metaheuristic were independently proposed in the eighties by Glover [68] and Hansen [84], and since then TS has been widely applied to combinatorial optimization problems. A comprehensive introduction to TS can be found in the book by Glover and Laguna [71], or in Hertz, Taillard and de Werra [87].

TS is essentially a sophisticated and improved type of *local search*, an algorithm that in its simplest form, also known as Hill Climbing, works as follows. Consider a starting current solution, evaluate its neighboring solutions (according to a given neighborhood structure), and set the best or the first found neighbor which is better than the current solution as new current solution. Iterate this process until an improving solution is found in the neighborhood of a current solution. The local search stops when the current solution is better than all its neighbors, that is, when the current solution is a *local optimum*.

Such a simple and very general local search behaves quite poorly in practice, particularly because when a local optimum is found, the algorithm stops improving, and combinatorial problems often have local optima whose objective values are much worse than that of the global optimum. The strength of the TS metaheuristic with respect to simple local search is that, by employing three TS-specific concepts, it avoids to get prematurely stuck in a local optimum. These TS-specific concepts are: *best improvement*, *tabu lists*, and *aspiration criteria*.

Best improvement means that each current solution is always replaced by its best neighbor, even if the best neighbor is worse than the current solution. This is clearly a way not to get stuck in local optima. Using best improvement poses the problem of possible cycling among already visited solutions, because it is possible, for example, that the best neighbor of a solution is indeed the last visited current solution. In order to avoid cycling, choosing recently visited solutions is forbidden, by storing some attributes of these solutions in the so-called *tabu lists*. Whole solutions are not stored in a tabu list, because this would require too much memory for most combinatorial optimization problems. The

choice of attributes is a delicate point. Typically, tabu lists store the 'moves' that should be performed in order to go from one solution to another, or the differences between solutions. In this way the memory requirement of tabu lists is feasible, but another problem arises: forbidding all solutions corresponding to a tabu attribute may forbid also solutions that have not yet been visited, and possibly also very good or optimal solutions. TS employs *aspiration criteria* for solving this problem. An aspiration criterion is a condition that, if satisfied, allows to set as new current solution a solution obtained by performing a tabu move. A typical example of aspiration criterion is requiring that a solution is better than the best solution found from the beginning of the algorithm.

In pseudocode, the TS metaheuristic may be represented as in Algorithm 6, where x, y are feasible solutions of the combinatorial optimization problem,

Algorithm 6 Tabu Search (TS)

```
Generate a starting current solution x
Initialize the tabu lists
for iteration k = 1, 2, ... do
  Set A(x, k) = {y ∈ S(x)\T(x, k) ∪ T̃(x, k)}
  Set x = arg min_{y∈A(x,k)} g(y)
  Update the tabu lists and the aspiration criteria
end for
```

$A(x,k)$ is the set of solutions among which the new current solution is chosen at iteration k, $S(x)$ is the set of neighbors of x, $T(x,k)$ is the set of tabu moves at iteration k, and $\tilde{T}(x,k)$ is the set of tabu moves satisfying at least one aspiration criterion. In TS, typical stopping criteria may be a maximum CPU time, a maximum number of consecutive iteration not producing an improving solution, or the emptiness of the set $A(x,k)$.

Theoretical properties of convergence of TS to the optimal solutions has been analyzed only quite recently by Hanafi [82] and by Glover and Hanafi [70]. Both papers derive convergence results for a version of TS where the choice of a given neighborhood and a decision criterion for selecting moves force some solutions to be revisited before exploring other new ones.

4.4.1 TS for SCOPs

In comparison to the wide literature about TS for DCOPs, there are still very few papers applying TS to SCOPs. These works are all experimental, and address static SCOPs both in the case of exact objective and ad-hoc approximation, and sampling approximation.

Exact objective or ad-hoc approximation As we have already pointed out, one of the major difficulties when solving SCOPs is that the objective function, even if explicitly computable, is computationally expensive. In local search algorithms, TS included, it is crucial to be able to evaluate the neighborhood of a solution efficiently. Therefore, one of the main issues of applying

TS to SCOPs is indeed to find efficient approximations of the objective value difference between couples of neighboring solutions.

Gendreau et al. [66] propose a TS algorithm for solving the vehicle routing problem with stochastic demands and customers. One of the major contribution of their paper is indeed the development of an easily computed approximation for the objective function, used for the evaluation of potential moves. The proposed TS was quite successful in experiments: for instances up to about 50 customes, it was able to find optimal solutions in about 90% of cases, with an average deviation of 0.38% from optimality.

Other papers applying TS to SCOPs are the already cited [21, 22] by Bianchi et al. (see section 4.1, 4.2, and 4.3), where a simple TS algorithm has been compared with other metaheuristics (ACO, EC, SA, and Iterated Local Search). Similarly to the other metaheuristics, two approximations for the objective value difference between neighboring solutions generated according to the OrOpt scheme have been tested, the Yang and the TSP approximation. Even if the two approximations have different characteristics (the first one is more accurate but more computationally expensive than the second), the quality of results produced by the two versions of TS seemed to be quite insensitive to the type of approximation. In [22], TS obtained results better than ACO and SA, but worse than EC.

Sampling approximation In the literature two types of contributions may be distinguished: papers that inside TS use simulation as a black box for the evaluation of the objective value of solutions, and papers that adapt the simulation procedure to the different components of TS, such as neighborhood exploration, setting of tabu moves, verification of aspiration criteria, in order to speed up the computation.

To the first group belong the papers by Lutz et al. [104], Finke et al. [56], Dengiz and Alabas [45]. These papers apply quite standard TS techniques, and are usually very time consuming, since the evaluation of solutions by simulation is a time consuming process often relying on extern or commercial simulation packages. The advantage of using simulation is that in this way the real objective function is considered, in problems where a rigorous mathematical programming formulation would impose severe unrealistic restrictions.

Among the second group of papers (adapting simulation to the different components of TS), we describe in some detail the works by Costa and Silver [44] and by Aringhieri [8]. Costa and Silver [44] describe a TS algorithm for a problem in the context of cause-effect analysis, where the true cause of an undesirable effect must be recognized and eliminated. Given that the time to investigate a cause is a random variable with known probability distribution, the goal is to establish a fixed sequence of n causes so as to maximize the expected reward associated with discovering the true cause within a specified time horizon. This problem is also called stochastic ordering problem with time constraint (SOPTC).

The TS developed in this context, called NTS (Noisy TS), is based on sam-

pling and statistical tests, and is suited for all optimization problems where the evaluation of the objective function is computationally expensive due to the presence of noise in the problem definition. In the following we only describe the characteristics of NTS that are directly related to the stochastic nature of the problem. What we do not describe, is part of standard TS techniques for permutation problems. The objective value of a new current solution is computed by a sample average of the type of eq. (13). N samples are generated according to the so-called descriptive sampling technique as described in [95], in order to obtain substantial variance reduction with respect to other sampling methods. Descriptive sampling has been adopted by Costa and Silver also because in this way the quality of estimation of the exact objective value does not depend on the quality of the pseudo-random generator used. The estimation of the objective function takes $O(Nn)$ time, and if N is large enough to guarantee a good estimation quality, this computation may be quite time consuming. For this reason, the evaluation of the (possible many) neighbors of a solution is done with the following method relying on a smaller number of samples. A statistical test is used to decide whether a considered neighbor $y_c \in A(x,k)$ is better than the best neighbor $y_b \in A(x,k)$ examined so far in the current iteration k. The decision is done in two phases. First, a small number $N_c < N$ of samples is randomly generated for estimating the expected value $g_{N_c}(y_c)$ of y_c. The decision as to wether the true objective value of y_c, is higher than that of y_b is done by hypothesis testing. Second, if the test 'has decided' that y_c is better than y_b, this is further ascertained, by using all the N samples. If it results that $g_N(y_c) > g_N(y_b)$, than y_b is replaced by y_c. Since N is finite, notwithstanding the use of this double-check procedure, there is a certain probability that y_b is not truly the best feasible neighbor, and that the best solution so far is updated with not the truly best solution so far. In order to lesser the risk of missing a very good solution due to the bad quality of sampling, NTS keeps track of the ns best solutions encountered so far. At the end of the run all solutions in this list are re-evaluated with a number of samples $\kappa > N$, and the best solution according to this new estimation is the solution returned by NTS.

In [44], the influence on the performance NTS of several factors has been experimentally analyzed: the hypothesis testing technique (the t-test, the Wilcoxon test, and the Median test have been compared), and the number of samples N, N_c and κ to be used in the different phases of NTS. NTS has been compared also with a TS that is similar in everything to NTS, except for the fact that the objective function is computed exactly on the base of a closed form expression available for SOPTC, and no hypothesis test is performed. TS outperforms NTS both in computation time and solution quality, but the solution quality is only slightly better than NTS. This is a result that encourages the use of NTS for problems with very complex, or impossible to compute, objective functions. Note, however, that when a closed form expression for the objective function is available, even if it is quite computationally expensive like in SOPTC, it may still be more efficient to use the classical TS algorithm, instead of NTS.

An application where sampling is employed to save time with respect to using the exact objective function is the one by Aringhieri [8], that applies TS

to a Chance Constrained Program (Definition 3). Constraints are supposed to be linear functions, that is, in Definition 3 we pose $H_i(x,\omega) = \sum_j a_{ij}x_j - b_i(\omega)$, with $j = 1, 2 \ldots, n$ and $x \in S \subset \mathbb{R}^n$. Note that in this problem, only the vector b is assumed to be random. In the proposed TS, sampling is used to estimate the probability $p_i(x,k)$ that at iteration k solution x violates constraint b_i. Given a set of Nm random samples $b_{i,r}$, $i = 1, 2, \ldots, m$, $r = 1, 2, \ldots N$, the probabilities are estimated as follows

$$p_i(x,k) = \frac{\sum_{r=1}^{N} \delta_{i,r}}{N}, \quad \text{where } \delta_{i,r} = \begin{cases} 1 & \text{if } \sum_j a_{ij}x_j - b_{i,r} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

Probabilities $p_i(x,k)$ are used to define the concept of *probably tabu* moves that in practice extends the set of tabu moves. A move is *probably tabu* at iteration k, if it leads to a solution x for which $p_i(x,k) > \alpha_i$, $i = 1, 2, \ldots, m$ (compare this with eq. (3)). Given the set $P(x,k)$ of probably tabu neighbors of x, the new TS, called SIMTS-CCP (simulation TS for Chance Constrained Programs), can be obtained from algorithm 6 by modifying the computation of $A(x,k)$ as

$$A(x,k) = \{y \in S(x) \backslash T(x,k) \backslash P(x,k) \cup \tilde{T}(x,k)\}. \tag{15}$$

In [8] the SIMTS-CCP algorithm has been applied to two NP-hard optimization problems arising in the design of telecommunication networks. Preliminary computational results show that solution quality is comparable to that obtained by a TS algorithm that addresses the problem as deterministic, and the increase in computation time is acceptable.

4.5 Stochastic Partitioning Methods

We have grouped under the name of SPM the Beam Search heuristic applied to SCOPs [13, 55], the Stochastic Branch and Bound [109, 110], and the combined procedure inspired by Nested Partitions [114]. These methods, explicitly designed for SCOPs, follow in different ways the same search strategy: the search space is recursively partitioned in sub-spaces, and the computation effort is concentrated on the sub-spaces that are estimated to be the most promising ones. SPM are not usually considered as belonging to the class of metaheuristics, but they could, since inside the general search strategy, several heuristics may be employed for the evaluation of search sub-spaces, for the improvement of solutions, and for the estimation and comparison among solutions. In the following, we introduce the different SPM methods in the context of the type of SCOP that each method mainly focus on.

Exact objective or ad-hoc approximation The Beam Search (BS) heuristic is a heuristic strategy closely related to Branch and Bound, where the search space is recursively partitioned in sub-spaces, for which upper and lower bounds for the objective function are computed, in order to guide the search in the more promising partitions. Unlike Branch and Bound, BS reduces the width of the search moving downward in the search tree only from a limited number of best

promising nodes. The success of BS depends on the evaluation function that is used to select the nodes that will be further explored. Typically, in BS different evaluation functions are used. First, a simple but imprecise evaluation function is used at to discard some nodes (this phase is called *filtering*); second, nodes that survive filtering are subject to a more precise and time consuming evaluation. Thus, the main principles behind BS (partitioning the search space and dosing the computation effort in specific partitions) are similar to those of SBB and NP. The BS has been only recently applied to SCOPs. Beraldi and Ruszczyński [13] consider chance constrained problems like the ones we described in section 3.1. They apply BS to a set covering problem with probabilistic constraints, and show experimentally that BS allows a considerable time saving with respect to an exact Branch and Bound algorithm, and the solution quality of BS goes from optimal to 5% worse than optimal. Erel, et al. [55] present a BS-based method for the stochastic assembly line balancing problem in U-lines. Computational experiments indicate that the average performance of the proposed method is better than the best-known heuristic in the literature for the traditional straight-line problem.

Sampling approximation Stochastic Branch and Bound (SBB) has been first proposed by Norkin, Ermoliev, and Ruszczyński [109], as a method for solving problems where the objective function must be estimated by sampling as described in section 3.3.2. This algorithm extends to SCOPs the main principle of the classical Branch and Bound, that is, the computation of upper and lower bounds for the objective function of portions of the search space, in order to guide the search. The main difference with respect to classical Branch and Bound is that here, due to the stochastic and non-exact estimation of the objective function (and thus of the upper and lower bounds), sub-spaces cannot in general be cut during the search, but a sort of backtracking into previously evaluated sub-spaces may be necessary.

The SBB algorithm proposed in [109] is represented by the pseudocode of Algorithm 7, and works as follows. Given the search space S, the algorithm constructs increasingly finer partitions of S, denoted by $\mathcal{P} = \{S^1, S^2, \ldots\}$. The original problem of finding $g^*(S) := min_{x \in S}\{g(x)\}$ (see eq. (2)), is divided into the sub-problems of finding $g^*(S^r) := min_{x \in S^r}\{g(x)\}$, with $r = 1, 2, \ldots$, and $g^*(S) = min_{S^r \in \mathcal{P}}\{g^*(S^r)\}$. Assume that there exist functions L and U from $\mathcal{P}$ to $\mathbb{R}$ such that, for each $S^r \in \mathcal{P}$, $L(S^r) \leq g^*(S^r) \leq U(S^r)$, and $U(S^r) = g(\bar{x})$ for some $\bar{x} \in S^r$, and if S^r is a singleton set, then $L(S^r) = g^*(S^r) = U(S^r)$. Suppose that the lower and upper bounds $L(S^r)$ and $U(S^r)$ cannot be exactly computed, but instead estimates $\lambda^l(S^r)$ and $\nu^m(S^r)$ are used, respectively, assuming that almost surely $\lim_{l \to \infty} \lambda^l(S^r) = L(S^r)$, and $\lim_{m \to \infty} \nu^m(S^r) = U(S^r)$.

Norkin et al. [109] proved the following convergence result: Suppose the indices l_k and m_k are chosen in such a way that whenever a subset S^r is an element of $\mathcal{P}_k$ for infinitely many k, then $\lim_{k \to \infty} l_k = \infty$ and $\lim_{k \to \infty} m_k = \infty$. Then with probability one there exists an iteration k_0 such that for all $k \geq k_0$, the lowest-bound subsets $\bar{S}_k$ are singletons and contain optimal solutions only.

Algorithm 7 Stochastic Branch and Bound (SBB)

1: Set $\mathcal{P}_0 = S$, $\lambda_0(S) = \lambda^{l_0}(S)$, $\nu_0(S) = \nu^{m_0}(S)$;
2: **for** iteration $k = 0, 1, 2, \ldots$ **do**
3: Select the lowest-bound subset $\bar{S}_k \in \operatorname{argmin}_{S^r \in \mathcal{P}_k}\{\lambda_k(S^r)\}$ and a current solution $x^k \in \operatorname{argmin}_{S^r \in \mathcal{P}_k}\{\nu_k(S^r)\}$;
4: **if** the lowest-bound subset $\bar{S}_k$ is a singleton **then**
5: $\mathcal{P}_{k+1} = \mathcal{P}_k$;
6: **else**
7: Construct a partition of the lowest-bound subset $\mathcal{P}'_k(\bar{S}^k) = \{\bar{S}^k_1, \bar{S}^k_2, \ldots, \bar{S}^k_{n_k}\}$;
8: Construct a new full partition $\mathcal{P}_{k+1} = \mathcal{P}_k \backslash \{\bar{S}_k\} \cup \mathcal{P}'_k(\bar{S}^k)$;
9: **end if**
10: **for all** subsets $S^r \in \mathcal{P}_k$ **do**
11: Update the estimates of lower and upper bounds $\lambda_k(S^r) = \lambda^{l_k}(S^r)$, $\nu_k(S^r) = \nu^{m_k}(S^r)$;
12: **end for**
13: **end for**

As suggested in [109], the estimation of a lower bound $L(S^r)$ for $g^*(S^r)$, may be done by exchanging the minimization and the expectation operator, since $g^*(S^r) = \min_{x \in S^r} g(x) = \min_{x \in S^r} \mathbb{E}_P(G(x, \omega)) \geq \mathbb{E}_P\big(\min_{x \in S^r} G(x, \omega)\big)$. Thus, one may chose $L(S^r) = \mathbb{E}_P\big(\min_{x \in S^r} G(x, \omega)\big)$, and the estimation of the lower bound $L(S^r)$ may be computed by the sample average

$$\lambda^N(S^r) = \frac{1}{N} \sum_{j=1}^{N} \min_{x \in S^r} G(x, \omega_j), \tag{16}$$

where $\omega_1, \omega_2, \ldots, \omega_N$ is an independent, identically distributed (i.i.d.) random sample of N realizations of the random vector ω.

In general, the practical application of SBB implies one major difficulty: computing an estimation of the lower bound by eq. (16) requires solving a possibly NP-hard deterministic combinatorial optimization problem, $\min_{x \in S^r} G(x, \omega_j)$, for every sample scenario ω_j, and this is unfeasible in a reasonable amount of computation time, unless very small problem instances are addressed.

Gutjahr et al. [76] use SBB to solve small instances of the single-machine-tardiness scheduling problem. They consider different sampling techniques for estimating lower bounds, and report computational experiments.

As a way to make SBB more efficient, Gutjahr et al. [79] propose to use heuristics or metaheuristics to approximately solve the deterministic subproblems for the lower bound estimation of eq. (16), as schematized by Figure 3. The authors focus on the problem of Activity Crashing in Project Management, and show experimentally that the replacement of an exact solution to deterministic subproblems by a heuristic one (in this case a local search algorithm) is very advantageous. The authors also say that it is possible to extend the

convergence results of [109] to cases in which the deterministic subproblems are approximately solved by a search heuristic with a random starting point, keeping track of the best solution found so far. Another practical enhancement of the SBB proposed in [79] is the use of Importance Sampling as a technique to reduce the variance of the sample average estimates. Without the use of a variance-reduction technique, the number of Monte Carlo samples (and thus of computation time) required to obtain a sample average with the same variance would be much greater.

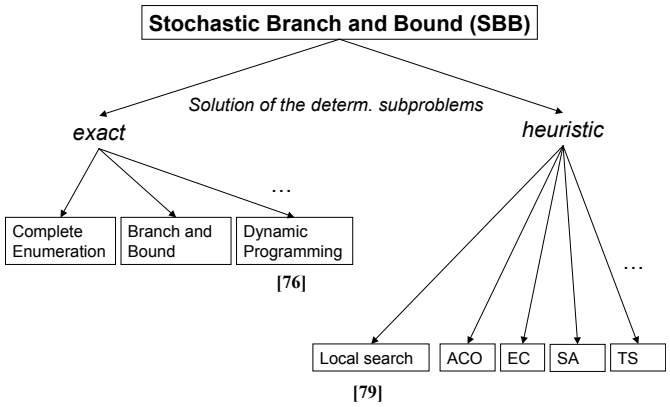

Figure 3: Possible ways of solving the deterministic subproblems for the computation of the lower bound (eq. (16)) in SBB.

Pichitlamchen and Nelson [114] propose a combined procedure extending the Nested Partitions (NP) method by Shi and Ólafsson [127] to SCOPs where the objective is estimated by sampling. NP is based on identifying a sequence of 'most promising' subsets of the search space S, and concentrating the search of good solutions there. At each iteration, the most promising subset of S is partitioned into M subsets, and the entire surrounding region is aggregated into one single subset of S. Thus, at each iteration NP looks at a partition of $M+1$ subsets of the search space S. From each of these $M+1$ subsets, a random solution is chosen using some random sampling scheme, and the objective value of each solution is evaluated, in order to decide which is the most promising subset of the next iteration. With respect to NP, the combined procedure of Pichitlamchen and Nelson applied to SCOPs includes a number of enhancements. First, in each of the current $M+1$ subsets, more than one solution is randomly chosen for evaluating the most promising subset. Second, solutions here are evaluated by the sample average estimation the objective value (see eq. (13)). Moreover, in order to select the best solution of each subset, and the best solution of all subsets, a statistical procedure called Sequential Selection with Memory (SMM) is used. SMM guarantees to select the best or near-best alternative among a set of solutions with a user-specified probability. It also

exploits memorized information (samples and sample averages) on previously encountered solutions. The spirit of SMM is similar to the F-Race procedure proposed in the context of ACO [27] (see section 4.1), since it consists in a series of steps in which a set of competing solutions is evaluated and the worst of them are eliminated. For details about SMM, see also [112, 113]. The combined procedure based on NP also applies a Hill Climbing local search (HC) to the best solution of each iteration. In this way, the computational effort is concentrated on the most promising subset of the search space. Another specific characteristic of the combined procedure of Pichitlamchen and Nelson is that at the end of the algorithm, the solution having the smallest sample average accumulated over all visits to that solution is returned as final solution. Pichilamchen and Nelson call their combined procedure NP+SMM+HC, a name that underlines its main building blocks just described. In [114], they provide a proof that, with probability one, NP+SMM+HC finds one of the optimal solutions as the number of iterations goes to infinity. Moreover, numerical experiments applying the algorithm to an (s,S) Inventory Problem and to a Three-Stage Buffer allocation problem show that NP+SMM+HC has a good performance in comparison to a pure random search and to a SA algorithm. While the convergence guarantee of NP+SMM+HC is due to the global guidance system provided by NP, the practical performance is enhanced by the use of SMM selection-of-the-best method and HC local search.

4.6 Other algorithmic approaches to SCOPs

In this section we briefly give some references to other approaches for solving SCOPs, such as Progressive Hedging and Rollout Algorithms, that we have not reviewed in the previous sections because too little developed in the literature. Nevertheless, we include this section because the referenced approaches may be either classified as metaheuristics, or they may involve metaheuristics as part of their solution strategy.

Progressive Hedging (PH) is an algorithm proposed by Rockafellar and Wets [119] for solving multistage stochastic programs. It is based on considering a set of few representative scenarios that capture the uncertain future; for each of these scenarios, a deterministic optimization subproblem is solved; in this way one ends up with more solutions, neither of which is in general feasible for the original problem. Therefore, a sort of averaging procedure among the solutions of the subproblems is performed, in order to obtain a 'blended' solution that hedges against future uncertainty. Some extensions of PH involve the use of heuristics or metaheuristics to solve the deterministic subproblems. For example, Løkketangen and Woodruff [103] integrate TS in the PH framework and apply this combined procedure to mixed integer (0,1) multistage stochastic programming. Haugen et al. [85] propose an extension of PH that is explicitly proposed as a metaheuristic: rather that using a heuristic algorithm to solve deterministic subproblems, it uses an algorithm for subproblems that is exact in its usual context, but severs as a heuristic for the proposed PH metaheuristic.

Rollout algorithms (RO) are an emerging class of methods for solving combi-

natorial optimization problems, that are capable of improving the performance of a given heuristic through sequential applications. Originally, the ideas underlying RO have been proposed by Tesauro and Galperin in 1997 [135] for developing a simulation-based algorithm to play blackgammon. In the same year, Bertsekas et al. [18] formalized RO for combinatorial optimization problems, by applying them to a machine maintainance and repair problem. RO are based on the Policy Iteration algorithm, which is part of the Dynamic Programming framework for solving MDP [14] (see the paragraph about Markov Decision Processes of section 3.2). Some authors (Bertsekas et al. in Section 2 of [18], and Bertsekas on page 528 of [15]), emphasize that RO also share some ideas with Tabu Search, particularly with the sequential fan candidate list strategy (Glover and Laguna [71]) and its extended variant, the fan and filter method (Glover [69]). Among the papers that apply RO to SCOPs, we cite the works of Secomandi on the vehicle vouting problem with stochastic demands [124, 125] and on the TSP with stochastic travel times [126], and the paper by Bertsekas and Castañon [16] on stochastic scheduling.

5 Discussion

This section takes a transversal view on the reviewed metaheuristics, and points out the main open issues and possible directions of research.

Using the simulation approximation We have seen that the selection-of-the-best method that a metaheuristic uses for performing sample averages and for comparing solutions can have a great impact on the effectiveness of the algorithm, but it is still hard to say which method is the most effective in relation to the metaheuristic where it is employed, and this is an interesting open issue.

Table 4 reports some successful selection-of-the-best methods described in the previous sections in the context of the metaheuristic where they have been used. In some cases ([74, 77, 6, 5, 89]), the use of a particular method has been justified mainly by the need to derive rigorous properties of convergence, and the application to other metaheuristics is not very meaningful. But in more experimental oriented papers, a method which is particularly efficient in one metaheuristic, could be advantageous also in others. This is the case, for instance, of F-Race [27], SMM [114], and the adaptive sampling procedures used in [75], [90], and [44]. In looking for efficient selection-of-the-best methods to be applied in metaheuristics, the literature about statistical procedures of ranking, selection, and multiple comparisons (see, for example, [59, 132] and the references cited therein) could be a good source of inspiration. Moreover, for speeding up the sample average computations, it could be useful the application of variance-reduction techniques, such as, for example, those belonging to the field of Rare Event Simulation [122].

Given the above observations, a selection-of-the-best method working as a black box simulation that does not allow to specify how samples are chosen

is not advisable. Another requirement that seems necessary is the possibility to increase the accuracy of objective function estimates, particularly when the algorithm has identified good or near optimal solutions. The intuitive reason is that often in SCOPs there are many local optima, whose values may be also quite near, and in order to discriminate between local optima one needs that the estimation error is small with respect to the difference between the exact value of local optima. We have seen a practical confirmation of this in several experimental papers, for instance [139] and [44]. A more rigorous argument in favor of this requirement is that all metaheuristics with provable convergence properties need to use a number of samples increasing with the iteration counter.

It has been recognized that completely different discrete stochastic optimization algorithm may be needed for small and for large search spaces, respectively (cf. [59]). Also the "degree of randomness", that is, the size of noise compared to the undisturbed objective function values, is an important factor. It cannot be expected that a metaheuristic variant working well for large solution spaces with small noise will also perform optimally for small solution spaces with a large amount of noise, and vice versa. It appears that a characterization of metaheuristic variants for SCOPs with respect to their appropriate domains of problem instance types still waits for being elaborated.

Experimental comparisons among different metaheuristics At the moment, most of the papers in the SCOP literature focus on one single metaheuristic, which is compared either to variants of the same metaheuristic, or to simple heuristics such as random search, or to exact methods when these are available. Only a very small number of papers perform comparisons among different metaheuristics, as reported in Table 5. Thus, it is still impossible to give guidelines on which metaheuristic is better in which situation.

One important aspect that experimentation with different metaheuristics could reveal is whether the effectiveness of a metaheuristic is due to the particular adjustments to speed up the computation (like approximating the objective function or using carefully designed sampling and statistical comparison strategies), or to the intrinsic search trajectory of the metaheuristic.

Theoretical convergence properties Papers analyzing theoretical convergence properties of metaheuristics applied to SCOPs are summarized in Table 6. Note that in the table SCOPs with exactly computable objective function are missing. In fact, when a metaheuristic always uses an exact expression for the objective value of a solution, its convergence behavior is equivalent, from a theoretical point of view, to that of applying the metaheuristic to a DCOP (pointers to theoretical analyses of metaheuristics for DCOPs have been provided in the previous sections while introducing each metaheuristic). On the

Reference(s)	Selection-of-the-best method		Metaheuristic(s) where the method is used
	Way of evaluating a solution	Solutions compared	
Gutjahr [74]	Sample average with number of samples increasing linearly with the iteration counter	Current solution with current estimation of optimal solution	ACO
Gutjahr [75]	Sample average with number of samples decided adaptively on the base of a statistical test	Current solution with current estimation of optimal solution	ACO, SA
Birattari et al. [27]	Sample averages integrated with the F-Race statistical procedure	All solutions belonging to the (dynamic) set of solutions in the race	ACO
Gutjahr and Pflug [77]	Sample average with number of samples increasing more than quadratically with the iteration counter	Current solution with current estimation of optimal solution	SA
Alrefaei and Andradóttir [6]	Sample average and normalized number of visits	All solutions visited so far	SA
Alkhamis et al. [5]	Sample average with number of samples increasing with iterations, comparison with a statistical significance test	Current solution with current estimation of optimal solution	SA
Homem-de-Mello [89, 90]	Sample average with number of samples decided adaptively on the base of a t-test	Current solution with current estimation of optimal solution	SA
Costa and Silver [44]	Descriptive sampling, statistical test in two stages (using a higher number of samples only if first stage of the test is positive)	Current solution with current estimation of optimal solution, keeping in memory a given number of good solutions for final, more accurate comparison	TS
Gutjahr et al. [76]	Sample average using the Importance Sampling variance-reduction technique, and number of samples increasing with the iteration counter	Lower and upper bounds of all subsets in which the search space has been partitioned	SBB (SPM)
Pichitlamchen and Nelson [114]	Sample averages integrated with the SMM statistical procedure	Random selected solutions in each subset in which the search space has been partitioned, and random solutions selected from the neighborhood of the current solution during local search	NP+SMM+HC (SPM)

Table 4: Main selection-of-the-best methods used in the metaheuristics literature for SCOPs where the objective function must be estimated by sampling.

Reference	SCOP	Metaheuristics compared	"Winner"
Bianchi et al. [21, 22]	VRPSD	ACO, EC, SA, TS, ILS	EC, TS
Gutjahr [75]	TSPTW	ACO, SA	ACO
Pichitlamken and Nelson [114]	Inventory Problem and Three-stage Buffer Allocation Problem	SPM, SA	SPM
Easton and Mansour [53]	Stochastic Goal Programming	EC, SA, TS	EC

Table 5: Papers with comparisons among different metaheuristics.

contrary, when ad-hoc approximations of the objective function are used, the systematicness of the error makes a theoretical analysis very difficult.

Theoretical convergence analyses do exist for static SCOPs with sampling estimated objective function. The most studied metaheuristic from this point of view is certainly SA, followed by ACO and SPM. TS and EC still miss this kind of analysis. Interstingly, Homem-de-Mello [90] suggests that the results he derives for a simple variable-sample random search algorithm can be readily adapted to show the convergence of variable-sample versions of more sophisticated methods, in particular, those methods for which the proof of convergence in the DCOP domain relies on the convergence of pure random search. The author indicates explicitly EC (GA) as one of these, by referring to the work of Rudolph [123].

Finally, metaheuristics with provable convergence properties (ACO and EC) have been designed to solve MDPs. Actually, at the moment MDPs have been addressed *only* theoretically by metaheuristics. Therefore, there is the need to validate their effectiveness also by experimental investigations.

6 Conclusion

In this paper, a wide class of combinatorial optimization problems under uncertainty addressed by metaheuristics is considered. The domain of Stochastic Combinatorial Optimization Problems (SCOPs) is clearly identified by providing the formal description of several SCOPs. Metaheuristics that have been applied so far to SCOPs are introduced and the related literature is throughly reviewed. In particular, the following properties of metaheuristics have emerged from this survey: they are a valid alternative to exact classical methods for addressing real-sized SCOPs; they are flexible, since they can be quite easily adapted to solve different SCOPs formulations, both static and dynamic. The main open issues that need further investigation are: given a particular SCOP, which is the most promising metaheuristic to be applied; given a particular SCOP, which is the best way to use objective function approximations in the

Metaheuristic	SCOP category	Referece(s)
ACO	sampling estimated objective	Gutjahr [74]
SA	“ ”	Alrefaei and Andradóttir [6]
SA	“ ”	Alkhamis et al. [5]
SA	“ ”	Homem-de-Mello [89]
SA	“ ”	Alkhamis and Ahmed [4]
SBB (SPM)	sampling estimated objective	Norkin et al. [109]
SA	objective function subject to normally distributed noise	Gelfand and Mitter [63]
SA	objective function subject to noise reducing to zero after a certain number of iterations	Fox and Heine [58]
SA	objective function subject to noise distributed according to a sufficiently ‘peaked’ distribution	Gutjahr and Pflug [77]
ACO	infinite horizon MDP	Chang et al. [42] and Chang [41]
EC	infinite horizon MDP	Chang et al. [43]
EC	finite horizon partially observed MDP	Lin et al. [102]

Table 6: Papers with theoretical convergence proofs.

optimization process. As far as this last point, this survey highlights some methods that are less promising than others, providing some guidelines and references for future research.

Acknowledgments

The authors would like to thank Nicola Secomandi for the useful suggestions and informations he provided during the writing phase of the paper.

References

[1] E. Aarts and J. Korst. *Simulated Annealing and the Boltzmann Machine*. John Wiley & Sons, New York, NY, USA, 1990.

[2] A. N. Aizawa and B. W. Wah. Scheduling of genetic algorithms in a noisy environment. *Evolutionary Computation*, 2:97–122, 1994.

[3] S. Albers. Online algorithms: A survey. *Mathematical Programming*, 97(1-2):3–26, 2003.

[4] T. M. Alkhamis and M. A. Ahmed. Simulation-based optimization using simulated annealing with confidence intervals. In R. G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, editors, *Proceedings of the 2004*

Winter Simulation Conference (WSC04), pages 514–518. IEEE Press, Piscataway, NJ, USA, 2004.

[5] T. M. Alkhamis, M. A. Ahmed, and W. Kim Tuan. Simulated annealing for discrete optimization with estimation. *European Journal of Operational Research*, 116:530–544, 1999.

[6] M. H. Alrefaei and S. Andradóttir. A simulated annealing algorithm with constant temperature for discrete stochastic optimization. *Management Science*, 45:748–764, 1999.

[7] S. Andradóttir. A review of simulation optimization techniques. In D. J. Medeiros, E. F. Watson, J. S. Carson, and M. S. Manivannan, editors, *Proceedings of the 1998 Winter Simulation Conference (WSC98)*, pages 151–158. IEEE Press, Piscataway, NJ, USA, 1998.

[8] R. Aringhieri. Solving chance-constrained programs combining tabu search and simulation. In C. C. Ribeiro and S. L. Martins, editors, *Proceedings of the 3rd International Workshop on Experimental and Efficient Algorithms (WEA04)*, volume 3059 of *Lecture Notes in Computer Science*, pages 30–41. Springer, Berlin, Germany, 2004.

[9] D. Arnold. In *Noisy Optimization with Evolutionary Strategies*, volume 8 of *Genetic Algorithms and Evolutionary Computation Series*. Kluwer Academic Publishers, Boston, MA, USA, 2002.

[10] J.-F. M. Barthelemy and R. T. Haftka. Approximation concepts for optimum structural design - a review. *Structural Optimization*, 5:129–144, 1993.

[11] T. Beielstein and S. Markon. Threshold selection, hypothesis tests, and DOE methods. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002)*, volume 1, pages 12–17. IEEE Press, Piscataway, NJ, USA, 2002.

[12] R. Bellman and L. A. Zadeh. Decision-making in a fuzzy environment. *Management Science*, 17:141–161, 1970.

[13] P. Beraldi and A. Ruszczyński. Beam search heuristic to solve stochastic integer problems under probabilistic constraints. *European Journal of Operational Research*, 167(1):35–47, 2005.

[14] D. P. Bertsekas. *Dynamic Programming and Optimal Control, Volumes 1 and 2*. Athena Scientific, Belmont, MA, USA, 1995.

[15] D. P. Bertsekas. *Network Optimization: Continuous and Discrete Models*. Athena Scientific, Belmont, MA, USA, 1998.

[16] D. P. Bertsekas and D. A. Castañon. Rollout algorithms for stochastic scheduling problems. *Journal of Heuristics*, 5:89–108, 1998.

[17] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-dynamic Programming*. Athena Scientific, Belmont, MA, USA, 1996.

[18] D. P. Bertsekas, J. N. Tsitsiklis, and C. Wu. Rollout algorithms for combinatorial optimization. *Journal of Heuristics*, 3(3):245–262, 1997.

[19] H.-G. Beyer. Evolutionary algorithms in noisy environments: Theoretical issues and guidelines for practice. *Computer Methods in Applied Mechanics and Engineering*, 186(2-4):239–267, 2000.

[20] H.-G. Beyer, E. Brucherseifer, W. Jakob, H. Pohlheim, B. Sendhoff, and T. Binh To. Evolutionary algorithms - terms and definitions. `http://ls11-www.cs.uni-dortmund.de/people/beyer/EA-glossary/def-engl-html.html`.

[21] L. Bianchi, M. Birattari, M. Chiarandini, M. Manfrin, M. Mastrolilli, L. Paquete, O. Rossi-Doria, and T. Schiavinotto. Metaheuristics for the vehicle routing problem with stochastic demands. In X. Yao, E. Burke, J. A. Lozano, J. Smith, J. J. Merelo Guervós, J. A. Bullinaria, J. Rowe, P. Tiňo, A. Kabán, and H.-P. Schwefel, editors, *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, volume 3242 of *Lecture Notes in Computer Science*, pages 450–460. Springer, Berlin, Germany, 2004.

[22] L. Bianchi, M. Birattari, M. Manfrin, M. Mastrolilli, L. Paquete, O. Rossi-Doria, and T. Schiavinotto. Hybrid metaheuristics for the vehicle routing problem with stochastic demands. *Journal of Mathematical Modelling and Algorithms*. To appear.

[23] L. Bianchi and A. M. Campbell. Extension of the 2-p-opt and 1-shift algorithms to the heterogeneous probabilistic traveling salesman problem. *European Journal of Operational Research*. To appear.

[24] L. Bianchi, L. M. Gambardella, and M. Dorigo. An ant colony optimization approach to the probabilistic traveling salesman problem. In J. J. Merelo Guervós, P. Adamidis, H.-G. Beyer, J.-L. Fernández-Villacañas, and H.-P. Schwefel, editors, *Proceedings of the 7th International Conference on Parallel Problem Solving from Nature (PPSN-VII)*, volume 2439 of *Lecture Notes in Computer Science*, pages 883–892. Springer, London, UK, 2002.

[25] L. Bianchi, L. M. Gambardella, and M. Dorigo. Solving the homogeneous probabilistic traveling salesman problem by the ACO metaheuristic. In M. Dorigo, G. Di Caro, and M. Sampels, editors, *Proceedings of the 3rd International Workshop on Ant Algorithms (ANTS 2002)*, volume 2463 of *Lecture Notes in Computer Science*, pages 176–187. Springer, London, UK, 2002.

[26] L. Bianchi, J. Knowles, and N. Bowler. Local search for the probabilistic traveling salesman problem: correction to the 2-p-opt and 1-shift algorithms. *European Journal of Operational Research*, 162(1):206–219, 2005.

[27] M. Birattari, P. Balaprakash, and M. Dorigo. ACO/F-Race: ant colony optimization and racing techniques for combinatorial optimization under uncertainty. In R. F. Hartl et al., editor, *Proceedings of the 6th Metaheuristics International Conference (MIC 2005)*, 2005. To appear.

[28] J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer, New York, NY, USA, 1997.

[29] Z. W. Birnbaum. On random variables with comparable peakedness. *Annals of Mathematical Statistics*, 19:76–81, 1948.

[30] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308, 2003.

[31] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, Cambridge, MA, USA, 1998.

[32] N. E. Bowler, T. M. A. Fink, and R. C. Ball. Characterization of the probabilistic traveling salesman problem. *Physical Review E*, 68(036703), 2003.

[33] J. Branke and M. Guntsch. New ideas for applying ant colony optimization to the probabilistic TSP. In *Proceedings of the 3rd European Workshop on Evolutionary Computation in Combinatorial Optimization (EvoCOP 2003)*, volume 2611 of *Lecture Notes in Computer Science*, pages 165–175. Springer, Berlin, Germany, 2003.

[34] J. Branke and M. Guntsch. Solving the probabilistic TSP with ant colony optimization. *Journal of Mathematical Modelling and Algorithms*, 3(4):403–425, 2004.

[35] J. Branke and C. Schmidt. Selection in the presence of noise. In E. Cantú-Paz, J. A. Foster, K. Deb, D. Davis, R. Roy, U. M. O'Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. Dowsland, N. Jonoska, and J. Miller, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2003)*, pages 766–777. Springer, Berlin, Germany, 2003.

[36] M. Brusco and L. Jacobs. A simulated annealing approach to the cyclic staff-scheduling problem. *Naval Research Logistics*, 40(1):69–84, 1993.

[37] M. Brusco and L. Jacobs. A simulated annealing approach to the solution of flexible labour scheduling problems. *Journal of the Operational Research Society*, 44(12):1191–1200, 1993.

[38] A. A. Bulgak and J. L. Sanders. Integrating a modified simulated annealing algorithm with the simulation of a manufacturing system to optimize buffer sizes in automatic assembly systems. In M. Abrams, P. Haigh, and J. Comfort, editors, *Proceedings of the 1988 Winter Simulation Conference (WSC98)*, pages 684–690. IEEE Press, Piscataway, NJ, USA, 1988.

[39] P. Calégari, G. Coray, A. Hertz, D. Kobler, and P. Kuonen. A taxonomy of evolutionary algorithms in combinatorial optimization. *Journal of Heuristics*, 5:145–158, 1999.

[40] E. Cantú-Paz. Adaptive sampling for noisy problems. In K. Deb, R. Poli, W. Banzhaf, H.-G. Beyer, E. Burke, P. Darwen, D. Dasgupta, D. Floreano, J. Foster, M. Harman, O. Holland, P. L. Lanzi, L. Spector, A. Tettamanzi, D. Thierens, and A. Tyrrell, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2004)*, volume 3103 of *Lecture Notes in Computer Science*, pages 947–958. Springer, Berlin, Germany, 2004.

[41] H. S. Chang. An ant system based exploration-exploitation for reinforcement learning. In *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, pages 3805–3810. IEEE Press, Piscataway, NJ, USA, 2004.

[42] H. S. Chang, W. J. Gutjahr, J. Yang, and S. Park. An ant system approach to Markov decision processes. In *Proceedings of the 23rd American Control Conference (ACC04)*, volume 4, pages 3820–3825. IEEE Press, Piscataway, NJ, USA, 2004.

[43] H. S. Chang, H-G. Lee, M. Fu, and S. I. Marcus. Evolutionary policy iteration for solving Markov decision processes. *IEEE Transactions on Automatic Control*, 2005. To appear.

[44] D. Costa and E. A. Silver. Tabu search when noise is present: an illustration in the context of cause and effect analysis. *Journal of Heuristics*, 4:5–23, 1998.

[45] B. Dengiz and C. Alabas. Simulation optimization using tabu search. In J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, editors, *Proceedings of the 2000 Winter Simulation Conference (WSC00)*, pages 805–810. IEEE Press, Piscataway, NJ, USA, 2000.

[46] M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5(2):137–172, 1999.

[47] M. Dorigo and L. M. Gambardella. Ant Colony System: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1:53–66, 1997.

[48] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: an autocatalytic optimization process. Technical Report 91-016, Department of Electronics, Politecnico di Milano, Milan, Italy, 1991.

[49] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 26(1):29–41, 1996.

[50] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, Cambridge, MA, USA, 2004.

[51] M. Dorigo (European Project Coordinator). Metaheuristics network web site. `http://www.metaheuristics.org/`.

[52] M. Dyer and L. Stougie. Computational complexity of stochastic programming problems. Technical Report SPOR-report 2003-20, Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, Eindhoven, The Netherlands, 2003.

[53] F. Easton and N. Mansour. A distributed genetic algorithm for deterministic and stochastic labor scheduling problems. *European Journal of Operational Research*, 118(3):505–523, 1999.

[54] F. Easton and D. Rossin. A stochastic goal program for employee scheduling. *Decision Sciences*, 27(3):541–568, 1996.

[55] E. Erel, I. Sabuncuoglu, and H. Sekerci. Stochastic assembly line balancing using beam search. *International Journal of Production Research*, 43(7):1411–1426, 2005.

[56] D. A. Finke, D. J. Medeiros, and M. Traband. Shop scheduling using tabu search and simulation. In E. Yücesan, C. H. Chen, J. L. Snowdon, and J. M. Charnes, editors, *Proceedings of the 2002 Winter Simulation Conference (WSC02)*, pages 1013–1017. IEEE Press, Piscataway, NJ, USA, 2002.

[57] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial intelligence through simulated evolution*. John Wiley & Sons, New York, NY, USA, 1966.

[58] B. L. Fox and G. W. Heine. Probabilistic search with overrides. *Annals of Applied Probability*, 4:1087–1094, 1995.

[59] M. C. Fu. Optimization for simulation: Theory vs. practice. *INFORMS Journal on Computing*, 14(3):192–215, 2002.

[60] M. C. Fu. Guest editorial of the ACM TOMACS special issue on "simulation optimization". *ACM Transactions on Modeling and Computer Simulation*, 13(2):105–107, 2003.

[61] L. M. Gambardella and M. Dorigo. Solving symmetric and asymmetric TSPs by ant colonies. In *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation (ICEC'96)*, pages 622–627. IEEE Press, Piscataway, NJ, 1996.

[62] S. B. Gelfand and S. K. Mitter. Analysis of simulated annealing for optimization. In *Proceedings of the 24th IEEE Conference on Decision and Control (CDC'85)*, volume 2, pages 779–786. IEEE Press, Piscataway, NJ, USA, 1985.

[63] S. B. Gelfand and S. K. Mitter. Simulated annealing with noisy or imprecise measurements. *Journal of Optimization Theory and Applications*, 69:49–62, 1989.

[64] D. Geman and S. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. In *IEEE Transactions of Pattern Analysis and Machine Intelligence*, volume 6, pages 721–741, 1984.

[65] M. Gendreau, G. Laporte, and R. Séguin. An exact algorithm for the vehicle routing problem with stochastic demands and customers. *Transportation Sciences*, 29(2):143–155, 1995.

[66] M. Gendreau, G. Laporte, and R. Séguin. A tabu search heuristic for the vehicle routing problem with stochastic demands and customers. *Operations Research*, 44(3):469–477, 1996.

[67] M. Giacobini, M. Tomassini, and L. Vanneschi. Limiting the number of fitness cases in genetic programming using statistics. In J. J. Merelo Guervós, P. Adamidis, H.-G. Beyer, J.-L. Fernández-Villacañas, and H.-P. Schwefel, editors, *Proceedings of the 7th International Conference on Parallel Problem Solving from Nature (PPSN-VII)*, volume 2439 of *Lecture Notes in Computer Science*, pages 371–380. Springer, London, UK, 2002.

[68] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13:533–549, 1986.

[69] F. Glover. Future paths for integer programming and links to artificial intelligence. In J.-K. Hao, E. Lutton, E. Ronald, M. Schoenaurer, and D. Snyers, editors, *Artificial Evolution*, volume 1363 of *Lecture Notes in Computer Science*. Springer, Berlin, Germany, 1998.

[70] F. Glover. Tabu search and finite convergence. *Discrete Applied Mathematics*, 119:3–36, 2002.

[71] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[72] G. R. Grimmett and D. R. Stirzaker. *Probability and Random Processes, 3rd Edition*. Oxford University Press, New York, NY, USA, 2001.

[73] G. Gutin and A. Punnen, editors. *The Traveling Salesman Problem and its Variations*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.

[74] W. J. Gutjahr. A converging ACO algorithm for stochastic combinatorial optimization. In *Proceedings of the 2nd Symposium on Stochastic Algorithms, Foundations and Applicaions (SAGA 2003)*, volume 2827 of *Lecture Notes in Computer Science*, pages 10–25. Springer, Berlin, Germany, 2003.

[75] W. J. Gutjahr. S-ACO: An ant-based approach to combinatorial optimization under uncertainty. In *Proceedings of the 4th International Workshop on Ant Colony Optimization and Swarm Intelligence (ANTS 2004)*, volume 3172 of *Lecture Notes in Computer Science*, pages 238–249. Springer, Berlin, Germany, 2004.

[76] W. J. Gutjahr, A. Hellmayr, and G. Ch. Pflug. Optimal stochastic single-machine tardiness scheduling by stochastic branch-and-bound. *European Journal of Operational Research*, 117:396–413, 1999.

[77] W. J. Gutjahr and G. Ch. Pflug. Simulated annealing for noisy cost functions. *Journal of Global Optimization*, 8:1–13, 1996.

[78] W. J. Gutjahr, C. Strauss, and M. Toth. Crashing of stochastic activities by sampling and optimization. *Business Process Management Journal*, 6:65–83, 2000.

[79] W. J. Gutjahr, C. Strauss, and E. Wagner. A stochastic branch-and-bound approach to activity crashing in project management. *INFORMS Journal on Computing*, 12:125–135, 2000.

[80] J. Haddock and J. Mittenthal. Simulation optimization using simulated annealing. *Computers & Industrial Engineering*, 22:387–395, 1992.

[81] B. Hajek. Cooling schedules for optimal annealing. *Mathematics of Operations Research*, 13:311–329, 1988.

[82] S. Hanafi. On the convergence of tabu search. *Journal of Heuristics*, 7:47–58, 2000.

[83] W. K. K. Haneveld and M. H. van der Vlerk. Stochastic integer programming: state of the art. *Annals of Operations Research*, 85:39–57, 1999.

[84] P. Hansen. The steepest ascent mildest descent heuristics for combinatorial programming. Talk presented at the Congress on Numerical Methods in Combinatorial Optimization, Capri, Italy, 1986.

[85] K. K. Haugen, A. Lokketangen, and D. L. Woodruff. Progressive hedging as a meta-heuristic applied to stochastic lot-sizing. *European Journal of Operational Research*, 132:116–122, 2001.

[86] A. Hertz and D. Kobler. A framework for the description of evolutionary algorithms. *European Journal of Operational Research*, 126:1–12, 2000.

[87] A. Hertz, E. Taillard, and D. de Werra. Tabu search. In E. H. L. Aarts and J. K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 121–136. John Wiley & Sons, New York, NY, USA, 1997.

[88] J. H. Holland. *Adaptation in natural and artificial systems*. The University of Michigan Press, Ann Harbor, MI, USA, 1975.

[89] T. Homem-de-Mello. Variable-sample methods and simulated annealing for discrete stochastic optimization. Stochastic Programming E-Print Series, `http://hera.rz.hu-berlin.de/speps/`, 2000.

[90] T. Homem-de-Mello. Variable-sample methods for stochastic optimization. *ACM Transactions on Modeling and Computer Simulation*, 13:108–133, 2003.

[91] S. Irani, X. Lu, and A. Regan. On-line algorithms for the dynamic traveling repair problem. *Journal of Scheduling*, 7(3):243–258, 2004.

[92] O. Jellouli and E. Châtelet. Monte Carlo simulation and genetic algorithm for optimising supply chain management in a stochastic environment. In *Proceedings of the 2001 IEEE Conference on Systems, Man, and Cybernetics*, volume 3, pages 1835–1839. IEEE Press, Piscataway, NJ, USA, 2001.

[93] Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing*, 9(1):3–12, 2005.

[94] Y. Jin. Evolutionary optimization in uncertain environments - a survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317, 2005.

[95] H. Jönsson and E. A. Silver. Some insights regarding selecting sets of scenarios in combinatorial stochastic problems. *Journal of Production Economics*, 45:463–472, 1996.

[96] P. Kall and S. W. Wallace. *Stochastic Programming*. John Wiley & Sons, Chichester, UK, 1994. Wiley has released the copyright on the book, and the authors made the text available to the scientific community: it can be downloaded for free at `http://www.unizh.ch/ior/Pages/Deutsch/Mitglieder/Kall/bib/ka-wal-94.pdf`.

[97] A. Kenyon and D. P. Morton. A survey on stochastic location and routing problems. *Central European Journal of Operations Research*, 9:277–328, 2002.

[98] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[99] P. Kouvelis and G. Yu. *Robust Discrete Optimization and Its Applications*, volume 14 of *Nonconvex optimization and its applications*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1997.

[100] G. Laporte, F. Louveaux, and H. Mercure. An exact solution for the a priori optimization of the probabilistic traveling salesman problem. *Operations Research*, 42(3):543–549, 1994.

[101] Z.-Z. Lin, J. C. Bean, and C. C. White III. Genetic algorithm heuristics for finite horizon partially observed Markov decision processes. Technical Report 98-24, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI ,USA, 1998.

[102] Z.-Z. Lin, J. C. Bean, and C. C. White III. A hybrid genetic/optimization algorithm for finite-horizon, partially observed Markov decision processes. *INFORMS Journal on Computing*, 16(1):27–38, 2004.

[103] A. Løkketangen and D. L. Woodruff. Progressive hedging and tabu search applied to mixed integer (0,1) multistage stochastic programming. *Journal of Heuristics*, 2:111–128, 1996.

[104] C. M. Lutz, K. R. Davis, and M. Sun. Determining buffer location and size in production lines using tabu search. *European Journal of Operational Research*, 106:301–316, 1998.

[105] K. L. Mak and Z. G. Guo. A genetic algorithm for vehicle routing problems with stochastic demand and soft time windows. In M. H. Jones, S. D. Patek, and B. E. Tawney, editors, *Proceedings of the 2004 IEEE Systems and Information Engineering Design Symposium (SIEDS04)*, pages 183–190. IEEE Press, Piscataway, NJ, USA, 2004.

[106] S. Markon, D. V. Arnold, T. Bäck, T. Beielstein, and H.-G. Beyer. Thresholding – a selection operator for noisy ES. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2001)*, volume 1, pages 465–472. IEEE Press, Piscataway, NJ, USA, 2001.

[107] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[108] B. L. Miller and D. E. Goldberg. Genetic algorithms, selection schemes, and the varying effects of noise. *Evolutionary Computation*, 4(2):113–131, 1997.

[109] V. I. Norkin, Y. M. Ermoliev, and A. Ruszczyński. On optimal allocation of indivisibles under uncertainty. *Operations Research*, 46(3):381–395, 1998.

[110] V. I. Norkin, G. Ch. Pflug, and A. Ruszczyński. A branch and bound method for stochastic global optimization. *Mathematical Programming*, 83:425–450, 1998.

[111] S. Ólafsson and J. Kim. Simulation optimization. In E. Yücesan, C. H. Chen, J. L. Snowdown, and J. M. Charnes, editors, *Proceedings of the 2002 Winter Simulation Conference (WSC02)*, pages 89–84. IEEE Press, Piscataway, NJ, USA, 2002.

[112] J. Pichitlamken. *A combined procedure for optimization via simulation.* PhD thesis, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, USA, 2002.

[113] J. Pichitlamken and L. B. Nelson. Selection-of-the-best procedures for optimization via simulation. In B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, editors, *Proceedings of the 2001 Winter Simulation Conference (WSC01)*, pages 401–407. IEEE Press, Piscataway, NJ, USA, 2001.

[114] J. Pichitlamken and L. B. Nelson. A combined procedure for optimization via simulation. *ACM Transactions on Modeling and Computer Simulation*, 13(2):155–179, 2003.

[115] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* John Wiley & Sons, New York, NY, USA, 2005.

[116] M. Rauner, S. C. Brailsford, W. J. Gutjahr, and W. Zeppelzauer. Optimal screening policies for diabetic retinopathy using a combined discrete event simulation and ant colony optimization approach. In J. G. Andersen and M. Katzper, editors, *Proceedings of the 15th International Conference on Health Sciences Simulation, Western MultiConference 2005*, pages 147–152. SCS - Society of Computer Simulation International, San Diego, CA, USA, 2005.

[117] R. I. Rechenberg. *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der biologischen Evolution.* Frommann-Holzboog, Stuttgart, Germany, 1973.

[118] C. R. Reeves and J. E. Rowe. *Genetic Algorithms: Principles and Perspectives - a Guide to GA Theory.* Operaations Research/Computer Science Interfaces Series. Kluwer Academic Publishers, Boston, MA, USA, 2003.

[119] R. T. Rockafellar and R. J.-B. Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research*, 16:119–147, 1991.

[120] N. Roenko. Simulated annealing under uncertainty. Technical report, Institute for Operations Research, University of Zurich, Switzerland, 1990.

[121] S. L. Rosen and C. M. Harmonosky. An improved simulated annealing simulation optimization method for discrete parameter stochastic systems. *Computers & Operations Research*, 32(2):343–358, 2005.

[122] R. Y. Rubinstein. *Simulation and the Monte Carlo method*. John Wiley & Sons, New York, NY, USA, 1981.

[123] G. Rudolph. Convergence of evolutionary algorithms in general search spaces. In *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC'96)*, pages 50–54. IEEE Press, Piscataway, NJ, USA, 1996.

[124] N. Secomandi. Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands. *Computers & Operations Research*, 27(5):1171–1200, 2000.

[125] N. Secomandi. A rollout policy for the vehicle routing problem with stochastic demands. *Operations Research*, 49(5):796–802, 2001.

[126] N. Secomandi. Analysis of a rollout approach to sequencing problems with stochastic routing applications. *Journal of Heuristics*, 9:321–352, 2003.

[127] L. Shi and S. Ólafsson. Nested partitions method for global optimization. *Operations Research*, 48(3):390–407, 2000.

[128] P. Stagge. Averaging efficiently in the presence of noise. In A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature (PPSN-V)*, volume 1498 of *Lecture Notes in Computer Science*, pages 188–200. Springer, Berlin, Germany, 1998.

[129] Stochastic Programming Community Homepage. `http://stoprog.org/`.

[130] J. Sudhir Ryan Daniel and C. Rajendran. A simulation-based genetic algorithm for inventory optimization in a serial supply chain. *International Transactions in Operational Research*, 12(1):101–127, 2005.

[131] R. S. Sutton and A. G. Barto. *Reinforcement Learning*. The MIT Press, Cambridge, MA, USA, 1998.

[132] J. R. Swisher, S. H. Jacobson, and E. Yücesan. Discrete-event simulation optimization using ranking, selection, multiple comparison procedures: a survey. *ACM Transactions on Modeling and Computer Simulation*, 13(2):134–154, 2003.

[133] A. Teller and D. Andre. Automatically choosing the number of fitness cases: The rational allocation of trials. In J. R. Koza, D. Kalyanmoy, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Proceedings of the 2nd Annual Genetic Programming Conference (GP-97)*, pages 321–328. Morgan Kaufmann, San Francisco, CA, USA, 1997.

[134] D. Teodorović and G. Pavković. A simulated annealing technique approach to the vehicle routing problem in the case of stochastic demand. *Transportation Planning and Technology*, 16:261–273, 1992.

[135] G. Tesauro and G. R. Galperin. On-line policy improvement using monte carlo search. *Advances in Neural Information Processing Systems*, 9:1068–1074, 1997.

[136] J. N. Tsitsiklis. Asyncronous stochastic approximation and Q-learning. *Machine Learning*, 16:185–202, 1994.

[137] P. J. M. van Laarhoven and E. H. L. Aarts. *Simulated Annealing: Theory and Applications*. D. Reidel Publishing Company, Dordrecht, The Netherlands, 1987.

[138] M. Vose. *The Simple Genetic Algorithm: Foundations and Theory*. The MIT Press, Cambridge, MA, USA, 1999.

[139] J. P. Watson, S. Rana, L. D. Whitley, and Howe A. E. The impact of approximate evaluation on the performance of search algorithms for warehouse scheduling. *Journal of Scheduling*, 2(2):79–98, 1999.

[140] W. Yang, K. Mathur, and R. H. Ballou. Stochastic vehicle routing problem with restocking. *Transportation Science*, 34(1):99–112, 2000.

[141] M. Yokoyama and H. W. Lewis III. Optimization of the stochastic dynamic production problem by a genetic algorithm. *Computers & Operations Research*, 30:1831–1849, 2003.

[142] Y. Yoshitomi. A genetic algorithm approach to solving stochastic job-shop scheduling problems. *International Transactions in Operational Research*, 9(4):479–495, 2002.

[143] Y. Yoshitomi and R. Yamaguchi. A genetic algorithm and the Monte Carlo method for stochastic job-shop scheduling. *International Transactions in Operational Research*, 10(6):577–596, 2003.

[144] H. J. Zimmermann. *Fuzzy Set Theory and its Application*. Kluwer Academic Publishers, Boston, MA, USA, 2nd edition, 1991.