Available online at www.sciencedirect.com
SCIENCE DIRECT®

Robotics and Autonomous Systems 50 (2005) 99–114

Robotics and Autonomous Systems

www.elsevier.com/locate/robot

Non-communicative multi-robot coordination in dynamic environments

Jelle R. Kok*, Matthijs T.J. Spaan, Nikos Vlassis

Informatics Institute, Faculty of Science, University of Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, The Netherlands

Received 16 July 2003; accepted 6 August 2004
Available online 22 December 2004

Abstract

Within a group of cooperating agents the decision making of an individual agent depends on the actions of the other agents. In dynamic environments, these dependencies will change rapidly as a result of the continuously changing state. Via a context-specific decomposition of the problem into smaller subproblems, coordination graphs offer scalable solutions to the problem of multiagent decision making. In this work, we apply coordination graphs to a continuous (robotic) domain by assigning roles to the agents and then coordinating the different roles. Moreover, we demonstrate that, with some additional assumptions, an agent can predict the actions of the other agents, rendering communication superfluous. We have successfully implemented the proposed method into our *UvA Trilearn* simulated robot soccer team which won the RoboCup-2003 World Championship in Padova, Italy.

Keywords: Multiagent coordination; Coordination graphs; Game theory; RoboCup

1. Introduction

A multiagent (multi-robot) system is a group of agents that coexist in an environment and can interact with each other in several different ways in order to optimize a performance measure [1]. Research in multiagent systems aims at providing principles for the construction of complex systems containing multiple independent agents and focuses on behavior management issues (e.g., coordination of behaviors) in such systems.

We are interested in fully cooperative multiagent systems in which all agents share a common goal. A key aspect in such systems is the problem of coordination: the process that ensures that the individual decisions of the agents result in jointly optimal decisions for the group. In principle, game-theoretic techniques can be applied to solve the coordination problem [2], but this approach requires reasoning over the joint ac-

* Corresponding author. Tel.: +31 20 525 7524; fax: +31 20 525 7490.

E-mail addresses: jellekok@science.uva.nl (J.R. Kok), mtjspaan@science.uva.nl (M.T.J. Spaan), vlassis@science.uva.nl (N. Vlassis).

0921-8890/$ – see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2004.08.003

tion space of the agents, whose size is exponential in the number of agents. For practical situations involving many agents, modeling n-person games becomes intractable. However, the particular structure of the coordination problem can often be exploited to reduce its complexity.

A recent approach to decrease the size of the joint action space involves the use of a coordination graph (CG) [3]. In this graph, each node represents an agent, and an edge indicates that the corresponding agents have to coordinate their actions. In order to reach a jointly optimal action, a variable elimination algorithm is applied that iteratively solves the local coordination problems one by one and propagates the result through the graph using a message passing scheme. In a context-specific CG [4] the topology of the graph is first dynamically updated based on the current state of the world before the elimination algorithm is applied.

In this work we will describe a framework to coordinate multiple robots using coordination graphs. We assume a group of robotic agents that are embedded in a continuous and dynamic domain and are able to perceive their surroundings with sensors. The continuous nature of the state space makes the direct application of context-specific CGs difficult. Therefore, we appropriately 'discretize' the continuous state by assigning *roles* to the agents [5] and then, instead of coordinating the different agents, coordinate the different roles. It turns out that such an approach offers additional benefits: the set of roles not only allows for the definition of natural coordination rules that exploit prior knowledge about the domain, but also constrains the feasible action space of the agents. This greatly simplifies the modeling and the solution of the problem at hand.

Furthermore, we will describe a method that, using some additional common knowledge assumptions, allows an agent to predict the optimal action of its neighboring agents, making communication unnecessary. Finally, we work out an extensive example in which we apply coordination graphs to the RoboCup simulated soccer domain.

The setup is as follows: in Section 2 we review the coordination problem from a game-theoretic perspective, and in Section 3 we explain the concept of a coordination graph. In Section 4 we will describe our framework to coordinate agents in a continuous dynamic environment using roles without using communication. This is followed by an extensive example in the RoboCup soccer simulation domain in Section 5. We end with our conclusions and discuss possible further extensions in Section 6.

	thriller	comedy
thriller	1, 1	0, 0
comedy	0, 0	1, 1

Fig. 1. An example coordination game.

2. The coordination problem

In order to place the coordination problem in a broader context, we will first review it from a game-theoretic point of view. A strategic game [2] is a tuple $\langle n, \mathcal{A}_{1,\ldots,n}, R_{1,\ldots,n}\rangle$, where n is the number of agents, $\mathcal{A}_i$ the set of actions of agent i and R_i the payoff function for agent i. This payoff function maps the selected *joint* action $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$ to a real value: $R_i(\mathcal{A}) \to \mathbb{R}$. Each agent independently selects an action from its action set, and then receives a payoff based on the actions selected by all agents. The goal of the agents is to select, via their individual decisions, the most profitable joint action.

In this work we are interested in fully cooperative strategic games, so-called coordination games, in which all agents share the same payoff function $R_1 = \cdots = R_n = R$. Fig. 1 shows a graphical representation of a coordination game between two agents. The rows and columns correspond to the possible actions of, respectively, the first and second agent, while the entries contain the returned payoff for the corresponding joint action. Without knowing the choice of the other agent, each agent can choose between two types of movies, either a thriller or a comedy. The agents have to coordinate their actions in order to maximize their payoff since choosing the same movie results in a payoff of 1 for both agents, while choosing a different movie will provide them zero payoff.

A fundamental solution concept in strategic games is the Nash equilibrium [2,6]. It defines a joint action $a^* \in \mathcal{A}$ with the property that for every agent i holds $R_i(a_i^*, a_{-i}^*) \geq R_i(a_i, a_{-i}^*)$ for all actions $a_i \in \mathcal{A}_i$, where a_{-i} is the joint action for all agents excluding agent i. Such an equilibrium joint action is a steady state from which no agent can profitably deviate given the actions of the other agents. For example, the strategic game in Fig. 1 has two Nash equilibria corresponding to the situations where both agents select the same type of movie.

Another fundamental concept is Pareto optimality. An action a^* is Pareto optimal if there is no other joint action a for which $R_i(a) \geq R_i(a^*)$ for all agents i and $R_j(a) > R_j(a^*)$ for at least one agent j. That is, there is no other outcome that makes every player at least as well off and at least one player strictly better off. There are many examples of strategic games where a Pareto optimal solution is not a Nash equilibrium and vice versa (e.g., in the famous prisoner's dilemma [2]). However, in coordinated games such as the one depicted in Fig. 1 each Pareto optimal solution is also a Nash equilibrium by definition.

Formally, the coordination problem can be seen as the problem of selecting one single Pareto optimal Nash equilibrium[1] in a coordination game [1]. This can be accomplished using several different methods [7]: using communication, learning, or by imposing social conventions. In the first case an agent can inform the other agent of its action, restricting the choice of the other agents to a simplified coordination game. If in the movie example the first agent would notify the other agent that it will select the comedy, the coordination game is simplified to the second row which contains only one equilibrium. Secondly, learning can be used when the strategic game is played repeatedly. Each agent makes predictions about the actions of the other players based on the previous interactions and chooses its action accordingly. This approach has received much attention over the past several years [7–9]. Finally, social conventions are constraints on the action choices of the agents. It can be regarded as a rule to select one of all the possible equilibria. As long as this convention is common knowledge among the agents, no agent can benefit from not abiding it. This general, domain-independent method will always result in an optimal joint action and moreover, it can be implemented offline: during execution the agents do not have to explicitly coordinate their actions, e.g., via negotiation. For instance, we can create a lexicographic ordering scheme in which we first order the agents and then the actions in our previous example. Assuming the ordering '$1 \succ 2$' (meaning that agent 1 has priority over agent 2) and 'thriller $\succ$ comedy', the second agent can derive from the social conventions that the first agent will select the thriller and will therefore also choose the thriller.

In the above cases we assume that all equilibria can be found and coordination is the result of each individual agent selecting its individual action based on the same equilibrium. However, the number of joint actions grows exponentially with the number of agents, making it infeasible to determine all equilibria in the case of many agents. This calls for methods that first reduce the size of the joint action space before solving the coordination problem. One such approach, explained next, is based on the use of a coordination graph that captures local coordination requirements between agents.

3. Coordination graphs

In systems where multiple agents have to coordinate their actions, it is infeasible to model all possible joint actions since this number grows exponentially with the number of agents. Fortunately, most problems exhibit the property that each agent only has to coordinate with a small subset of the other agents, e.g., in many robotic applications only robots that are close to each other have to coordinate their actions. A recent approach to exploit such dependencies involves the use of a coordination graph (CG), which represents the coordination requirements of a system [3].

The main assumption is that the global payoff function $R(a)$ can be decomposed into a linear combination of local payoff functions, each involving only a few agents. For example, suppose that there are four agents and the following decomposition of the payoff function:

$$R(a) = f_1(a_1, a_2) + f_2(a_1, a_3) + f_3(a_3, a_4).$$

The functions f_i specify the local coordination dependencies between the actions of the agents and can be graphically depicted as in Fig. 2. A node in this graph represents an agent, denoted by A_i, while an edge defines a (possible directed) dependency between two agents. Only interconnected agents have to coordinate their actions at any particular instance. In the decomposition of $R(a)$, A_2 has to coordinate with A_1, A_4 has to coordinate with A_3, A_3 has to coordinate

[1] In the rest of this article, we denote a Pareto optimal Nash equilibrium simply by equilibrium, unless otherwise stated.

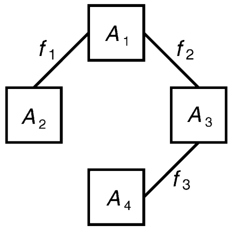


Fig. 2. An example coordination graph for a 4-agent problem.

with both A_4 and A_1, and A_1 has to coordinate with both A_2 and A_3. The global coordination problem is thus replaced by a number of local coordination problems each involving fewer agents.

In order to solve the coordination problem and find the optimal joint action a^* that maximizes $R(a)$, we can apply a variable elimination algorithm, which is almost identical to variable elimination in a Bayesian network [3,10]. The main idea is that the agents are eliminated one by one after performing a local maximization step which takes all possible action combinations of an agent's neighbors into account.

The algorithm operates as follows. One agent is selected for elimination, and it collects all payoff functions from its neighbors. Next, this agent optimizes its decision conditionally on the possible action combinations of its neighbors and communicates the resulting 'conditional' payoff function back to its neighbors. This conditional strategy is independent of this agent, which is then eliminated from the graph. The process continues with the next agent and ends when all but the last agent are eliminated. This agent simply chooses the individual optimal action that maximizes the final conditional strategy. A second pass in the reverse order is then needed so that all agents can determine their optimal action based on their conditional strategies and the fixed actions of their neighbors in the graph.

As an example, we first eliminate agent 1 in the aforementioned decomposition. This agent first collects the local payoff functions f_1 and f_2 and then only has to maximize over $f_1 + f_2$ in order to maximize $R(a)$. This can be written as

$$\max_a R(a) = \max_{a_2,a_3,a_4} \left\{ f_3(a_3, a_4) + \max_{a_1}[f_1(a_1, a_2) + f_2(a_1, a_3)] \right\}. \quad (1)$$

Agent 1 now creates a new payoff function $f_4(a_2, a_3) = \max_{a_1}[f_1(a_1, a_2) + f_2(a_1, a_3)]$ that returns the value corresponding to its best-response for the possible action combinations of agent 2 and agent 3. This function f_4 is independent of agent 1 and this agent is eliminated from the graph. The problem is now simplified to

$$\max_a R(a) = \max_{a_2,a_3,a_4} [f_3(a_3, a_4) + f_4(a_2, a_3)]. \quad (2)$$

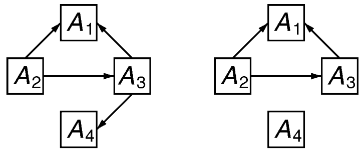
We now apply the same procedure to eliminate agent 2. In this case, only f_4 depends on the action of agent 2 and is replaced by the function $f_5(a_3) = \max_{a_2} f_4(a_2, a_3)$ producing

$$\max_a R(a) = \max_{a_3,a_4}[f_3(a_3, a_4) + f_5(a_3)] \quad (3)$$

which is independent of a_2. Next, we eliminate agent 3 by replacing the functions f_3 and f_5 with $f_6(a_4) = \max_{a_3}[f_3(a_3, a_4) + f_5(a_3)]$ giving $\max_a R(a) = \max_{a_4} f_6(a_4)$. Agent 4 now selects its optimal action given this propagated conditional strategy. A second pass in the reverse elimination order is performed in which each agent fixes it strategy based on its conditional strategy and the communicated actions from its neighbors. See [3,1] for more details.

The local payoff functions f_i can be matrix-based [3] as in Section 2 or rule-based [4]. In the latter case the payoff rules are defined using 'value rules', which specify how an agent's payoff depends on the current context. The context is defined as a propositional rule over the state variables and the actions of the agent's neighbors. These rules can be regarded as a sparse factorized representation of the complete payoff matrices since they are only specified for a context with non-zero payoff.

More formally, let $A_1, \ldots, A_n$ be a group of agents, where each agent A_j has to choose an action $a_j \in \mathcal{A}_j$ resulting in a joint action $a \in \mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$ and let X be a set of discrete state variables. The context c is then an element from the set of all possible combinations of the state and action variables, $c \in C \subseteq X \cup \mathcal{A}$. A value rule $\langle p; c : v\rangle \in P$ is a function $p : C \to \mathbb{R}$ such that $p(x, a) = v$ when $c = (x, a)$ is consistent with the current context and 0 otherwise. For a particular situation only those value rules contribute to the global payoff $R(a)$ that are consistent with the current context: $R(a) = \sum_{i=1}^{m} p_i(x, a)$ where m is the total number of value rules and x and a are, respectively, a state and joint action.



A_1	$\langle a_1 \wedge \overline{a_3} \wedge x : 4\rangle$	$\langle a_1 \wedge \overline{a_3} : 4\rangle$
	$\langle a_1 \wedge \overline{a_2} \wedge x : 5\rangle$	$\langle a_1 \wedge \overline{a_2} : 5\rangle$
A_2	$\langle \overline{a_2} \wedge x : 2\rangle$	$\langle \overline{a_2} : 2\rangle$
A_3	$\langle a_3 \wedge a_2 \wedge x : 5\rangle$	$\langle a_3 \wedge a_2 : 5\rangle$
A_4	$\langle a_3 \wedge a_4 \wedge \overline{x} : 10\rangle$	

Fig. 3. Initial coordination graph (left) and graph after conditioning on the context x = *true* (right).

As an example, consider the case where two persons have to coordinate their actions to pass through a narrow door. We describe this situation using the following value rule:

$$\begin{aligned}&\langle p_1;\ \text{in-front-of-same-door}(1, 2) \wedge \\ &\qquad a_1 = \text{passThroughDoor} \wedge \\ &\qquad a_2 = \text{passThroughDoor}: -50\rangle\end{aligned}$$

This rule indicates that when the two agents are located in front of the same door and both select the same action (passing through the door), the global payoff value will be reduced by 50. When the state is not consistent with the above rule (and the agents are not located in front of the same door), the rule does not apply and the agents do not have to coordinate their actions. By conditioning on the current state the agents discard all irrelevant rules, and as a result the CG is dynamically updated and simplified. Note that for the conditioning step each agent only needs to observe that part of the state mentioned in its own value rules.

For a more extensive example, see Fig. 3. Below the left graph all value rules, defined over binary action and context variables,[2] are depicted together with the agent the rule applies to. The coordination dependencies between the agents are represented by directed edges, where each (child) agent has an incoming edge from the (parent) agent that affects its decision. After the agents observe the current state, x = *true*, they condition on the context. The rule of A_4 does not apply and is removed. As a consequence, the optimal joint action is independent of the action of A_4 and the edge to A_4 is deleted from the graph as shown in the right graph of Fig. 3. In this case, A_4 can thus select either action without affecting the global reward $R(a)$.

After the agents have conditioned on the state variables, the agents are one by one eliminated from the graph. Let us assume that we first eliminate A_3 in the above example. Agent 3 first collects all rules from its children in which it is involved and then maximizes over the rules $\langle a_1 \wedge \overline{a}_3 : 4\rangle\langle a_3 \wedge a_2 : 5\rangle$. For all possible actions of A_1 and A_2, A_3 determines its best-response and then distributes the corresponding conditional strategy, in this case equal to $\langle a_2 : 5\rangle\langle a_1 \wedge \overline{a}_2 : 4\rangle$, to its parent A_2. Now a new directed edge from A_1 to A_2 is generated, since A_2 receives a rule containing an action of A_1. After this step, A_3 has no children in the coordination graph anymore and is eliminated. The procedure continues and after A_2 has distributed its conditional strategy $\langle a_1 : 11\rangle\langle\overline{a}_1 : 5\rangle$ to A_1, it is also eliminated. Finally, A_1 is the last agent left and fixes its action to a_1. Now a second pass in the reverse order is performed, in which each agent distributes its strategy to its parents, who then determine their final strategy. This results in the optimal joint action $\{a_1, \overline{a}_2, \overline{a}_3, a_4\}$ and a global payoff of 11. Note that $\{a_1, \overline{a}_2, \overline{a}_3, \overline{a}_4\}$ is also an optimal joint action. It depends on the individual action choice of A_4 which joint action is selected.

The outcome of the variable elimination algorithm is independent of the elimination order and the initial distribution of the rules and will always result in an optimal joint action [3]. However, the execution time of the algorithm does depend on the elimination order. In the table-based approach the cost of the algorithm is linear in the number of new functions introduced [3]. In the rule-based approach the cost is polynomial in the number of new rules generated in the maximization operation [4]. This number is never larger and often exponentially smaller than the complexity of the table-based approach.[3] Computing the optimal order for minimizing the mentioned runtime costs is known to be NP-complete [11], but good heuristics exists, e.g., minimum deficiency search which first eliminates the agents with the minimum difference between incoming and outgoing edges [12,13].

[2] The action a_1 corresponds to a_1 = *true* and the action $\overline{a}_1$ to a_1 = *false*.

[3] Do note that the rule-based approach involves an extra cost regarding the management of the sets of rules, causing its advantage to manifest primarily in problems with a fair amount of context specific dependencies [4].

A limitation of the described coordination approach is that it is based on propositional rules and therefore only applies to discrete domains. Next, we will show how to utilize this framework in continuous dynamic environments.

4. Dynamic continuous environments

We are interested in problems that involve multiple robots that are embedded in a continuous domain, have sensors with which they can observe their surroundings, and need to coordinate their actions. As a main example we will use the RoboCup simulation soccer domain [14]. The Robot Soccer World Cup (RoboCup) is an international research initiative that uses the game of soccer as a domain for artificial intelligence and robotics research. The *soccer server* [15] is the basis for the simulation competition. It provides a fully distributed dynamic multi-robot domain with both teammates and adversaries and models many real-world complexities such as noise in object movement, noisy sensors and actuators, limited physical ability and restricted communication. One team is represented by 11 different computer processes that independently interact with the simulator in order to fulfill their common goal of scoring more goals than their opponent. A graphical representation of the complete field modeled by the soccer server is shown in Fig. 4.

Depending on the current situation, certain agents on the field have to coordinate their actions, for example, the agent that controls the ball must coordinate with its surrounding team-mates in order to perform a pass and the defenders must coordinate how to position themselves with respect to each other to cover as much space as possible.

Fig. 4. Graphical representation of the soccer field.

Such dependencies can be modeled by a CG that satisfies the following requirements: (i) its connectivity should be dynamically updated based on the current (continuous) state, (ii) it should be sparse in order to keep the dependencies and the associated local coordination problems as simple as possible, (iii) it should be applicable in situations where communication is unavailable or very expensive.

In the remainder of this section, we will concentrate on the two main features of our proposed method, designed to fulfill the requirements mentioned above. The first is the assignment of roles to the agents in order to apply coordination graphs to continuous domains and to reduce the action sets of the different agents; the second is to predict the chosen action of the other agents, rendering communication superfluous.

4.1. Context-specificity using roles

Conditioning on a context that is defined over a continuous domain is difficult in the original rule-based CG representation. A way to 'discretize' the context is by assigning *roles* to agents [5,16,17,1]. Roles are a natural way of introducing domain prior knowledge to a multiagent problem and provide a flexible solution to the problem of distributing the global task of a team among its members. In the soccer domain for instance one can easily identify several roles ranging from 'active' or 'passive' depending on whether an agent is in control of the ball or not, to more specialized ones like 'striker', 'defender', 'goalkeeper', etc.

In [18] a role is defined as an abstract specification of the set of activities an individual or subteam undertakes in service of the team's overall activity. In our framework a role $m \in M$ defines this set of activities as a set of value rules P_m. In the original rule-based CG each agent has only one set of value rules, which then would have to include all rules for all roles. Now, based on the given role assignment only a subset of all value rules applies which simplifies the coordination graph. Furthermore, the value rules in P_m pose additional constraints on the role of other agents contained in the value rules, reducing the edges in the coordination graph even further. For instance, agent i in role 'goalkeeper' who controls the ball and considers to pass the ball to any of the agents j in role ‘defender’ could result in the following value rule:

$$\langle p_1^{\text{goalkeeper}}; \text{has-role-defender}(j) \wedge a_i = \text{passTo}(j) : 10 \rangle, \quad \forall j \neq i.$$

An assignment of roles to agents provides a natural way to parameterize a coordination structure over a continuous domain. The intuition is that, instead of directly coordinating the agents in a particular situation, we assign roles to the agents based on this situation and subsequently try to ‘coordinate’ the set of roles. The other roles $\{m'\} \in M$ mentioned in the set of value rules P_m for role m define a coordination subgraph structure on M. As such, the assigned roles induce a coordination graph between all agents, each executing a certain role.

A question which remains is how roles are assigned to agents. In this section we describe the communication based case, which we can exploit to use a distributed role assignment algorithm [5,16,1]. In the next section, we will concentrate on the situation in which communication is unavailable.

The role assignment algorithm, which is common knowledge among the agents, defines a sequence M' of roles where $|M'| \geq n$ which represents a preference ordering over the roles: the most ‘important’ role is assigned to an agent first, followed by the second most important role, etc. By construction, the same role can be assigned to more than one agent, but each agent is assigned only a single role. Each role m has an associated potential r_{im} which is a real-valued estimate of how appropriate agent i is for the role m in the current world state. These potentials r_{im} depend on features of the state space relevant for role m as observed by agent i. For example, relevant features for role ‘striker’ could be the time needed to intercept the ball or the global position on the field. Each agent computes its potential for each $m \in M'$ and sends these to the other agents. Now the first $m \in M'$ is assigned to the agent that has the highest potential for that role. This agent is no longer under consideration; the next m is assigned to another agent, and so on, until all agents have been assigned a role. This algorithm requires sending $O(|M|n)$ messages, as each agent has to send each other agent its potential r_{im} for all $m \in M'$.

The roles can be regarded as an abstraction of a continuous state to a discrete context, allowing the application of existing techniques for discrete-state CGs. Furthermore, roles can reduce the action space of the agents by ‘locking out’ specific actions. For example, the role of the goalkeeper does not include the action ‘score’, and in a ‘passive’ role the action ‘shoot’ is deactivated. Such a reduction of the action space can offer computational savings, but more importantly it can facilitate the solution of a local coordination game by restricting the joint action space to a subspace that contains less Nash equilibria.

4.2. Non-communicating agents

For the role assignment each agent has to communicate its potential for a certain role to all other agents. Furthermore, the variable elimination requires that each agent receives the payoff functions of its neighboring agents, and after computing its optimal conditional strategy communicates a new payoff function back to its neighbors. Similarly, in the reverse process each agent needs to communicate its decision to its neighbors in order to reach a coordinated joint action.

In many practical dynamic situations, the agents may not be able to communicate with all neighbors (or have the time to finalize all communication before selecting an action) due to failures or time constraints. However, when communication is unavailable the variable elimination algorithm can still be applied if we further impose the requirement that the payoff function of an agent i is common knowledge among all agents that are *reachable* from i in the CG. Since only agents that are reachable in the CG need to coordinate their actions, the above requirement in fact frees agents from having to communicate their local payoff functions during optimization.

The complete procedure is now as follows. In order to determine the role assignment, each agent computes in parallel the potential r_{im} that reflects how appropriate agent i is for the role $m = 1, \ldots, n$. This is done by calculating the potential r_{im} for all agents i located in its subgraph. Then the actual assignment of roles to agents is equal to the procedure described in Section 4.1. During the non-communicative role assignment, each agent calculates $O(|M|n)$ potentials and thus runs in time polynomial in the number of agents and roles. This in contrast to the communicating case where each agent only has to compute $O(|M|)$ potentials but in total $O(|M|n)$ potentials have to be communicated.

In a context-specific CG, each agent[4] performs an additional conditioning step using the state variables to simplify the graph structure. In the communication case, each agent has to observe those state variables that are represented in its own value rules. As a consequence, each agent also has to observe the state variables of the agents located in its subgraph when communication is unavailable.

In order to perform the variable elimination algorithm, agent i starts with eliminating itself and keeps removing agents until it computes its own optimal action unconditionally on the actions of the others. In the worst case, agent i needs to eliminate all agents $j \neq i$, for j reachable from i. Each agent thus runs the complete algorithm by itself in order to determine its own action. The main difference with the communicating case is with respect to the reverse pass. When multiple best-response actions contribute the same local value to the global payoff, the eliminated agent could choose (at random) which action to choose in the communication case. Without communication, we have to ensure that the different agents select the same action by imposing the additional constraint that the ordering in the actions sets of the agents is common knowledge.

In the non-communicative case the elimination order neither has to be fixed in advance nor has to be common knowledge among all agents as in [3], but each agent is free to choose *any* elimination order, for example, one that allows the agent to quickly compute its own optimal action. This is possible because a particular elimination order affects only the speed of the algorithm and not the computed joint action as described earlier.

In terms of complexity, the computational costs for each individual agent are clearly increased to compensate for the unavailable communication. Instead of only optimizing for its own action, in the worst case each agent has to calculate the action of every other agent in the subgraph. The computational cost per agent increases thus linearly with the number of new payoff functions generated during the elimination procedure. Communication, however, is not used anymore which allows for a speedup since these extra individual computations may now run in parallel. This is in contrast to the original CG approach where computations need to be performed sequentially.

In summary, we can apply the CG framework without using communication when all agents are able to run the same algorithm in parallel. For this, we have to make the following assumptions:

- the payoff functions of an agent i are common knowledge among all agents reachable from i,
- each agent i can compute the potential r_{im} for all agents i in its subgraph,
- the action ordering is common knowledge among all agents,
- for the context-specific CG all agents reachable from i also observe the state variables located in the value rules of agent i.

Finally, we note that the common knowledge assumption is strong and even in cases where communication is available it cannot always be guaranteed [19]. In multiagent systems without communication common knowledge can be guaranteed if all agents consistently observe the same world state, but this is also violated in practice due to partial observability of the environment (a soccer player has a limited field of view). In our case, when the agents have to agree on a particular role distribution given a particular context, the only requirement we impose is that the role assignment in a particular local context is based on those parts of the state that are, to a good approximation, fully observable by all agents involved in the role assignment. For example, in a soccer game the particular role assignment may require that a group of coordinating agents observe the position of each other in the field, as well as the positions of their nearby opponents, and have a rough estimate of the position of the ball (e.g., knowing that the ball is far away). As long as such a context is encountered, a local graph is formed which is disconnected from the rest of the CG and can be solved separately.

5. Experiments

We have applied the aforementioned framework in our simulation robot soccer team *UvA Trilearn* [20]. The main motivation was to improve upon the coordination during ball passes between team-mates. First of all, we have conducted an experiment in which a complete team strategy was specified using value rules

[4] Note that when we refer to an agent in the remainder of this section, we assume this agent is assigned a role and is coordinating with other roles (as discussed in the previous section).

and each player selected its action based on the result of the variable elimination algorithm. In this case we made the world fully observable for all agents and used no communication between the agents. Furthermore, we incorporated this framework in our competition team, which participated in the RoboCup-2003 World Championships. For this we made some necessary modifications since during competition the world is only partially observable. Both approaches will be explained next in more detail.

5.1. Fully observable, non-communicating team

In this section we will explain how we have constructed a complete team strategy using the value rules from the CG framework. We assume that the world is fully observable[5] such that each agent can model the complete CG algorithm by itself as explained in Section 4.2. This is necessary since the RoboCup soccer simulation does not allow agents to communicate with more than one agent at the same time, which makes it impossible to apply the original variable elimination algorithm. This has no effect on the outcome of the algorithm. Furthermore, we used the synchronization mode to ensure that the simulator only proceeds to the next cycle when all the actions of the players are received. This is to make sure no action opportunities are lost because of the computation time of the algorithm.

In order to accomplish coordination, all agents first perform the role assignment, which maps each agent to one of the following ordered sequence of roles

$$M' = \{\text{active, receiver, receiver, passive, passive,}$$
$$\text{passive, passive, passive, passive, passive}\}.$$

There are four possible roles. The first and most important role is that of the active player which performs an action with the ball. We distinguish between two different types of roles, interceptor and passer, depending whether the ball can be kicked by the active player or not. Next, we assign the two roles of possible ball receivers. The remaining seven players are assigned the role of passive player. Note that we disregard the goalkeeper for simplicity.

The corresponding potentials for each role are specified as follows:

- The potential $r_{i,\text{active}}$ for the active player is equal to $1/t_i$ where $t_i > 0$ is the predicted time it will take player i to intercept the ball. For this, we use the modification of Newton's method as described in [21]. This method finds the least root (equal to the first possible interception time) of the function that represents the difference between the traveled distance of the ball and the movement of the player i. The actual role, passer or interceptor, that is assigned to the agent depends on the relative distance to the ball. When the ball is close enough to be kicked, the agent is assigned the role of passer, otherwise the role of interceptor.
- The potential for the role of receiver is based on the relative distance $d_{i,\text{b}}$ to the ball and the relative distance to the opponent goal $d_{i,\text{g}}$ for player i.

$$r_{i,\text{receiver}} = \begin{cases} \frac{1}{\max(1,d_{i,\text{g}})} + 1 & \text{if } d_{i,\text{b}} < k, \\ \frac{1}{\max(1,d_{i,\text{g}})} & \text{otherwise.} \end{cases} \tag{4}$$

  This function simply states that there is a preference for agents that are located close to the opponent goal. Furthermore, an additional reward is given when the ball is within a range of $k = 28$ m to player i.[6] This has the effect that agents that are outside this range are only taken into consideration for passing when there is no nearby alternative.
- The potential $r_{i,\text{passive}}$ for the role of passive player is a constant such that all remaining agents are assigned to this role.

The role assignment function is recomputed after every new observation and as a consequence the graph structures changes dynamically as the state of the world changes. A common example of a role assignment is depicted in Fig. 5, where the agent with the ball is a passer, the two players in range of the passer are receivers and the other players are passive. This assignment of roles defines the structure of the coordination graph. By construction an agent in a passive role always performs the same individual action, namely, moving towards its strategic position. This drastically simplifies the coordination graph since there are no dependencies between the passive agents.

[5] This is a configuration setting in the soccer server.

[6] After this distance the ball has not much speed left when shot with maximal velocity.

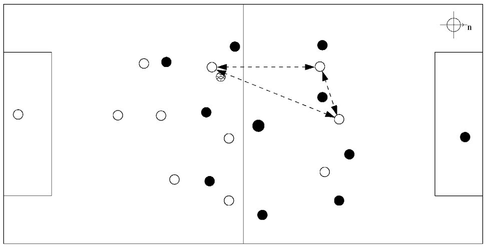


Fig. 5. A situation involving one passer and two possible receivers. The other agents are passive.

Now the coordination structure is known, all connected agents apply the elimination algorithm to determine their actions. For this, we have to define the value rules that consist of both state and action variables. First, we will give an overview of the different actions available to each agent:

- *passTo*(i, *dir*): pass the ball to a position with a fixed distance from agent i in the direction $dir \in D = \{center, n, nw, w, sw, s, se, e, ne\}$. The direction parameter specifies a direction relative to the receiving agent. 'North' is always directed toward the opponent goal and 'center' corresponds to a pass directly to the current agent position,
- *moveTo*(*dir*): move in the direction $dir \in D$,
- *dribble*(*dir*): move with the ball in direction $dir \in D$,
- *score*: shoot to the best spot in the opponent goal [22],
- *clearBall*: shoot the ball hard between the opponent defenders to the opponent side,
- *moveToStratPos*: move to the agent's strategic position based on its home position and the position of the ball which serves as an attraction point.

All mentioned actions are available in the released parts of the source code of our *UvA Trilearn 2003* team.[7] A more detailed description of how these actions are transformed into primary actions is given in [20]. We also defined (boolean) state variables that extract important (high level) information from the world state:

- *is-pass-blocked*(i, j, *dir*) indicates whether a pass from agent i to agent j is blocked by an opponent or not. The actual position to which is passed is the position at a small fixed distance from agent j in direction *dir*. A pass is blocked when there is at least one opponent located within a cone from the passing player to this position.
- *is-empty-space*(i, *dir*), indicates that there are no opponents within a small circle in the specified direction *dir* of agent i.
- *is-in-front-of-goal*(i) indicates whether agent i is located in front of the opponent goal.

Using these action and state variables, we can define the complete strategy of our team by means of value rules, which specify the contribution to the global payoff in a specific context. The value rules are specified for each player i and make use of the above defined actions and state variables.[8]

$$
\begin{aligned}
&\langle p_1^{\text{interc.}} ;\ \text{intercept} : 10\rangle\\
&\langle p_2^{\text{passer}} ;\ \text{has-role-receiver}(j) \wedge\\
&\qquad \neg\text{isPassBlocked}(i, j, dir) \wedge\\
&\qquad a_i = \text{passTo}(j, dir) \wedge\\
&\qquad a_j = \text{moveTo}(dir) : u(j, dir) \in [5, 7]\rangle\ \forall j \neq i\\
&\langle p_3^{\text{passer}} ;\ \text{is-empty-space}(i; n) \wedge\\
&\qquad a_i = \text{dribble}(n) : 2\rangle\\
&\langle p_4^{\text{passer}} ;\ a_i = \text{clearBall} : 0.1\rangle\\
&\langle p_5^{\text{passer}} ;\ \text{is-in-front-of-goal}(i) \wedge\\
&\qquad \text{is-ball-kickable}(i) \wedge\\
&\qquad a_i = \text{score} : 10\rangle\\
&\langle p_6^{\text{passer}} ;\ \text{has-role-interceptor}(j) \wedge\\
&\qquad \neg\text{isPassBlocked}(j; i; dir) \wedge\\
&\qquad a_j = \text{intercept} \wedge\\
&\qquad a_i = \text{moveTo}(dir) : u(i, dir) \in [5, 7]\rangle\ \forall j \neq i\\
&\langle p_7^{\text{receiver}} ;\ \text{has-role-receiver}(k) \wedge\\
&\qquad \neg\text{isPassBlocked}(k, i, dir) \wedge\\
&\qquad a_j = \text{passTo}(k, dir2) \wedge\\
&\qquad a_k = \text{moveTo}(dir2) \wedge\\
&\qquad a_i = \text{moveTo}(dir) : u(i, dir) \in [5, 7]\rangle\ \forall j, k \neq i\\
&\langle p_8^{\text{receiver}} ;\ \text{moveToStratPos} : 1\rangle\\
&\langle p_9^{\text{passive}} ;\ \text{moveToStratPos} : 1\rangle
\end{aligned}
$$

[7] This fully documented source code release is freely available from our website: http://www.science.uva.nl/~jellekok/robocup/.

[8] Note that we enumerate all rules using variables. The complete list of value rules is the combination of all possible instantiations of these variables. In all rules, $dir \in D$.

The first five rules are related to the action options for the active player. The first rule, p_1, indicates that intercepting the ball is the only option when performing the interceptor role. As a passer, there are several alternatives. Value rule p_2 represents an active pass to the relative direction *dir* of player *j* which can be performed when there are no opponents along that trajectory and the receiving agent will move in that direction to intercept the coming pass. The value that is contributed to the global payoff is returned by $u(j, dir)$ and depends on the position where the receiving agent *j* will receive the pass (the closer to the opponent goal the better). The next three rules indicate the other individual options for the active player: dribbling (we only allow forward dribbling), clearing the ball and scoring. Rule p_6 indicates the situation in which a receiver already moves to the position it expects the current interceptor to pass the ball to when it reaches the ball. Using the same principle, we can also create more advanced dependencies. For example, rule p_7 indicates that a receiver can already move to a position it will expect the receiver of another pass to shoot the ball to. Rule p_8 describes the situation in which a receiving player moves to its strategic position on the field. This action is only executed when it is not able to coordinate with one of the other agents, since it has only a small global payoff value. Finally, rule p_9 contains the single action option for a passive player that always moves to its strategic position.

When the nine basic rules are instantiated, the total number of value rules equals 204. We illustrate that even with such a rather small set of rules a complete (although simple) team strategy can be specified that makes explicit use of coordination. Furthermore, the rules are easily interpretable which makes it possible to add prior knowledge into the problem. Another advantage is that the rules are very flexible: existing rules can directly be added or removed. This makes it possible to change the complete strategy of the team when playing different kinds of opponents.

We will now look into an example of how the above rules are put into practice. The above rules contain a lot of context-dependencies represented in the state variables. In Fig. 5 we already simplified the coordination graph by assigning roles to the agents, if we now condition further on the specific state variables, we get the graph depicted in Fig. 6, corresponding to the following applicable value rules

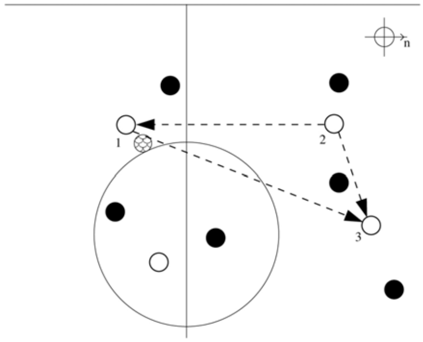


Fig. 6. The coordination graph in Fig. 3 after conditioning on the state variables. The passer (agent 1) decides to pass the ball to the first receiver (agent 2), while the second receiver (agent 3) moves to a good position for the first receiver to pass the ball to.

(we assume for simplicity that only the state variables ¬isPassBlocked(1, 2, s) and ¬isPassBlocked(2, 3, nw) are true):

$$
\begin{aligned}
A_1 : &\langle p_2^{\text{passer}} ; a_1 = \text{passTo}(2, s) \wedge \\
&\qquad a_2 = \text{moveTo}(s) : 6\rangle \\
&\langle p_3^{\text{passer}} ; a_1 = \text{dribble}(n) : 2\rangle \\
&\langle p_4^{\text{passer}} ; a_1 = \text{clearBall} : 0.1\rangle \\
A_2 : &\langle p_8^{\text{receiver}} ; a_2 = \text{moveToStratPos} : 1\rangle \\
A_3 : &\langle p_7^{\text{receiver}} ; a_1 = \text{passTo}(2, dir) \wedge \\
&\qquad a_2 = \text{moveTo}(dir) \wedge \\
&\qquad a_3 = \text{moveTo(nw)} : 5\rangle \ \forall\, dir \in D \\
&\langle p_8^{\text{receiver}} ; a_3 = \text{moveToStratPos} : 1\rangle
\end{aligned}
$$

Now the variable elimination algorithm can be performed. Each agent is eliminated from the graph by maximizing its local payoff. In the case that agent 1 is eliminated first, it gathers all value rules that contain a_1, maximizes its local payoff and distributes its conditional strategy consisting of the generated value rules

$$
\begin{aligned}
&\langle p_{10}^{\text{passer}} ; a_2 = \text{moveTo(s)} \wedge \\
&\qquad a_3 = \text{moveTo(nw)} : 11\rangle \\
&\langle p_{11}^{\text{passer}} ; a_2 = \text{moveTo(s)} \wedge \\
&\qquad a_3 = \neg\text{moveTo(nw)} : 6\rangle \\
&\langle p_{12}^{\text{passer}} ; a_2 = \neg\text{moveTo(s)} : 2\rangle
\end{aligned}
$$

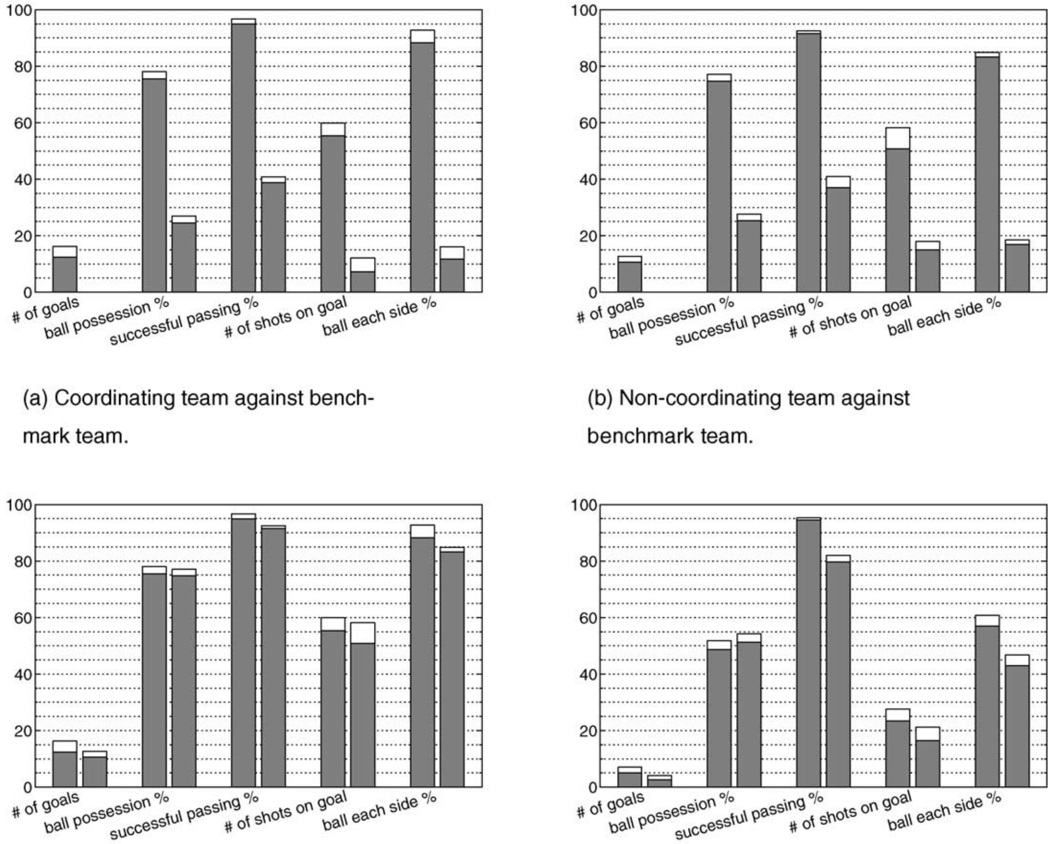
to its parents. Note that p_{10}^{passer} is formed by combining p_2^{passer} and p_7^{receiver} when both agents 2 and 3 fulfill the listed actions. When agent 3 performs a different action, the payoff is still 6 when agent 2 moves south as is stated in p_{11}^{passer}. When agent 2 also performs a different action, the only remaining action is the dribble with a payoff of 2. After agents 2 and 3 have also fixed their strategy, agent 1 will perform passTo(2, s), agent 2 will execute moveTo(s) to intercept the pass and agent 3 will perform moveTo(nw) to intercept a possible future pass of agent 2. During a match, this procedure is executed after each update of the state and the agents will change their action based on the new information. If in this example for some unpredicted reason the first pass fails, the graph will automatically be updated and correspond to the new situation.

To test this approach, we played games against ourselves, with one team using explicit coordination and the other team without using any coordination during passing. The latter case was modeled by deleting the rules p_6, p_7 from the list of value rules and removing the condition $a_j = \text{moveTo}(dir)$ from rule p_2 to indicate that it is not necessary for the receiver to anticipate the pass. Now, in the non-coordinating case a team-mate moves to the interception point only after he has observed a change in the ball velocity (after someone has passed the ball) and concludes that it is the fastest team-mate to the ball. Before the ball changes velocity, it has no notion of the fact that it will receive the ball and does not coordinate with the passing player. Furthermore, we also played matches against a benchmark version of our team in which the strategy was identical to our basic client implementation.[9] In this team the active player would intercept the ball and immediately kick it with maximal velocity to a random corner of the opponent goal. The implementation of the kick and the intercept are identical to the two other teams, with the result that the three teams only differ with respect to their (high level) strategy.

Since many different factors contribute to the overall performance of the team, it is difficult to measure the actual effect of the coordination with one single value. Therefore, we have concentrated on multiple statistics generated by the Statistics Proxy Server tool [23]. Fig. 7(a) and (b) show the game statistics for the coordinating and the non-coordinating team against the benchmark team averaged over 10 full-length games. These results show that both teams are able to defeat the benchmark team with considerable goal difference on all occasions (respectively $12.4 - 0$ and $10.6 - 0$). In Fig. 7(c) these statistics are directly compared with each other indicating that the coordinating team slightly outperforms the non-coordinating team. Fig. 7(d) shows the same statistics in the case the coordinating team plays against the non-coordinating team. In this setting, the coordinating team won 8 out of the 10 matches, drew one, and lost one. The average score was $5.5 - 2.6$. Almost all statistics show a performance improvement for the coordinating team. For example, the successful passing percentage was 94.55% for the team with the CG and 79.76% for the team without. These percentages indicate that due to the better coordination of the team-mates, fewer mistakes were made when the ball was passed between team-mates. This also has a positive effect on the other statistics, e.g., number of shots on goal and the location of the ball on the field. The only statistic in which the non-coordinating team (slightly) outperforms the coordinating one is in ball possession. After each failed attack the non-coordinating team has ball possession and can progress toward the opponent field without much difficulty. There are less attackers than defenders, so even when not explicitly coordinating it was possible to find a free team-mate to pass to. This all counted as ball possession for the non-coordinating team. However, when reaching the opponent side, it became more difficult to find a free team-mate because of the increased number of opponent players. As a result passing became more difficult and in many cases the ball was lost and the coordinating team could start their attack from the midfield. Since it is easier to keep ball possession when having to move the ball forward from your own backfield than to keep it when attacking from the midfield, the ball possession percentage is slightly higher for the non-coordinating team.

Table 1 shows the timing results for the different stages of the framework during the matches of the coordinating team against the non-coordinating team. For the non-coordinating team, the time to determine an action was approximately 3 ms in total for both the receiver and the passer. More time is needed for the coordinating team. This is mainly caused by the computation performed during the variable

[9] This is the same behavior as the released *UvA Trilearn* 2003 source code. Qualification for RoboCup-2003 was based on the performance of the teams against the 2002 version of this release.



(a) Coordinating team against benchmark team.

(b) Non-coordinating team against benchmark team.

(c) Statistics non-coordinating and coordinating team against benchmark team.

(d) Coordinating team against non-coordinating team.

Fig. 7. Mean and standard deviation of several statistics for the three tested teams. All results are averaged over 10 matches.

elimination algorithm. On average, the time needed for determining an action was 25.69 ms for the receivers. The time needed for computation is strongly related to the number of applicable value rules for a certain situation. On average approximately 25 value rules were applicable after conditioning on the context. However, situations occurred in which considerable more value rules were applicable. In these cases, the computation time also increased considerably. Fig. 8 shows the relationship between the number of value rules and the computation time. As the number of applicable value rules increases, the best-response function has to take more combinations of value rules into account, slowing down the computation.

5.2. Partially observable, non-communicating team

We participated in the RoboCup-2003 competition with our *UvA Trilearn* team. This team used the framework and value rules described in the previous section to determine its high-level strategy. However, in order to participate in the competition, we had to adapt the framework for (i) the partial observability of the domain and (ii) the real-time requirements of the simulator (100 ms per cycle).

The common knowledge assumption about the fully observable state cannot be made during competition since every agent only receives information of the part

Table 1
Average computation times (ms) for the different stages of the algorithm

Stage	Computation time (ms)	No. of samples
Initialization	7.15 (±4.45)	100
Role assignment	0.16 (±0.24)	602865
CG role passive	0.01 (±0.002)	472108
CG role interceptor	0.01 (±0.002)	45950
CG role receiver		
Condition step	3.94 (±4.74)	72307
Elimination step	34.87 (±72.15)	72307
No. of applicable rules	25.69 (±29.34)	72307
CG role active		
Condition step	4.15 (±4.61)	12500
Elimination step	48.93 (±96.83)	12500
No. of applicable rules	27.69 (±30.65)	12500

Results are generated on a 1.5 GHz computer with 512 MB. The results are combined for all players' actions over the course of 10 full-length games and are averaged over all players.

of the field to which its neck is oriented. However, the coordination graph structure specifies which parts of the state are relevant for coordination, i.e., the neighbors in the graph and their associated state variables. Therefore, we adjust the looking mechanism of the agents to actively orient their neck to the part of the field in which its neighbors in the graph are located and then assume that this part of the world is common knowledge among these agents. When all involved agents observe this information they can independently solve the local graph which is disconnected from the rest of the CG and so compensate for the missing state information. In our example, the passer and the receivers thus change their looking direction to their neighbors in the graph in order to get a good approximation of the relevant part of the state needed for coordination and are not interested in the passive players which are not connected to its subgraph.

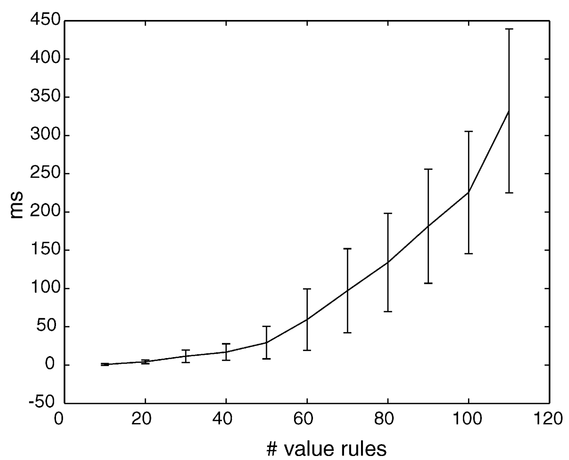


Fig. 8. The time (ms) needed for the elimination algorithm given the number of value rules that are applicable in the current context.

In order to comply with the real-time complexities of the simulator, the timing results of the previous section have to be improved. On average, a single player has approximately 10 ms in order to determine its action using our synchronization scheme [20]. Therefore, we included an additional preprocessing step during the conditioning in which for each of the nine basic rules and for each possible receiver only the value rule was kept that gave the highest reward. For example, a pass in the northern direction to a receiver has always precedence over a pass in southern direction and therefore we can remove the value rule related to the southern pass. In this case, we can afford to do so since we know that these coordinated passes are independent of the actions of the other agents. This reduces the number of value rules and makes sure that the agents are able to keep their computation time within the given constraints.

Finally, in order to improve the actual performed pass, we did not directly map the returned high-level actions from the CG algorithm to a primary action as in the previous section. Instead, given the returned receiver and direction, we evaluated multiple passes for different angles and different speeds in that direction using the modification of Newton's method [21] and selected the action that maximized the capture time between the receiver and the fastest opponent. This procedure can be regarded as a two-layered hierarchy in which a high-level action based on the global coordination situation is refined to a specific action which takes the local situation into account.

Using this approach, we were able to win the RoboCup-2003 World Championships in Padova, Italy, for which 46 teams had qualified. During the 16 matches played, *UvA Trilearn* had a total goal difference of 177 − 6. In the final we defeated TsinghuAeolus, the winner of 2001 and 2002, with a score of 4 − 3. The successful passing percentage was 91.43% for our team against 82.87% for TsinghuAeolus.

5.3. Related work

Two other coordination structures are Cohen and Levesque's joint intentions theory [24,25] and the SharedPlan theory [26]. The first is based on the notion that a joint action by a group of agents is more than the union of the simultaneous individual actions, even if those actions are coordinated. In order to act, a team must also be aware and care about the status of the group. A joint intention is a joint commitment to perform an action together. In contrast with joint intentions, the SharedPlan theory is not based on a joint mental state, but rather on two different kinds of intentions. An agent can intend to do some action or he can intend that some proposition holds. The first type is directed towards the agent's individual action, while the intention that is used for things like contemplating about sub-plans or helping teammates. The latter are defined via a set of axioms that specify for an agent which actions to take in order to facilitate its teammates, subteam or team to perform assigned tasks [26].

Tambe presents an implemented general model of flexible teamwork called STEAM [18]. This framework uses joint intentions as the basic building blocks of teamwork, but as in the SharedPlan theory team members build up a complex hierarchical structure of joint intentions, individual intentions and beliefs about other team member's intentions.

All three methods assume the agents form a team dynamically and communicate with each other extensively to negotiate with each other until the goal is reached. In [17], the notion of a locker-room agreement is introduced that facilitates coordination with little or no communication. This agreement provides a mechanism for pre-defining multiagent protocols that are accessible to the whole team, with the consequence that during execution the agents act autonomously, while still working towards a common goal. Furthermore, it is assumed that the agents can periodically synchronize their agreements during periods of unlimited and safe communication. This can be compared to our common knowledge (social conventions) assumptions about the value rules of the reachable agents in the graph, which make communication superfluous.

The notion of roles we use is similar to the ones described in [17,16,5]. In these settings, any agent has the knowledge about different roles which specify an agent's internal and external behaviors. Agents can at any time switch between the different roles based on external events or after negotiation. In these cases, coordination is the result of the different agents performing subtasks corresponding to their assigned role. In our case, the role assignment also defines the coordination structure, but next the CG framework is applied in order to coordinate the individual actions of the agents.

6. Conclusions and future work

We proposed two extensions to the framework of coordination graphs [3] for the cases where the agents are embedded in a continuous domain and/or communication is unavailable.

We assigned roles in order to abstract from the continuous state to a discrete context, allowing the application of existing techniques for discrete-state CGs. The role assignment specifies the coordination requirements between the different agents which is used by the CG framework in order to find the optimal combination of individual actions. This approach is based on value rules that specify the kind of coordination for a specific context. This approach is very flexible, since existing rules can directly be added or removed. This makes it possible to change the complete strategy of the team when playing different kinds of opponents.

Furthermore, we showed that we can dispense with communication if additional assumptions about common knowledge are introduced. This makes it possible to model the reasoning process of the other agents, making communication unnecessary.

Currently, the payoff values in the value rules are based on prior knowledge assumptions. As future work, we are interested in applying reinforcement learning techniques to a continuous-domain CG in order to learn the payoff functions in an automatic fashion.

Finally, from an application point of view we want to apply the CG model further to our team such that the agents also coordinate during other actions than passing, like organizing the defense or obstructing opponent passes. In order to accomplish this, we have to investigate how this method scales to bigger problems with a larger set of value rules.

Acknowledgement

This research is supported by PROGRESS, the embedded systems research program of the Dutch organization for Scientific Research NWO, the Dutch Ministry of Economic Affairs and the Technology Foundation STW, project AES 5414.

References

[1] N. Vlassis, A concise introduction to multiagent systems and distributed AI, Informatics Institute, University of Amsterdam, September 2003. http://www.science.uva.nl/~vlassis/cimasdai.

[2] M.J. Osborne, A. Rubinstein, A Course in Game Theory, MIT Press, 1994.

[3] C. Guestrin, D. Koller, R. Parr, Multiagent planning with factored MDPs, in: Advances in Neural Information Processing Systems, vol. 14, MIT Press, 2002, pp. 1523–1530.

[4] C. Guestrin, S. Venkataraman, D. Koller, Context-specific multiagent coordination and planning with factored MDPs, in: Proceedings of the 8th National Conference on Artificial Intelligence, Edmonton, Canada, 2002, pp. 253–259.

[5] M.T.J. Spaan, N. Vlassis, F.C.A. Groen, High level coordination of agents based on multiagent Markov decision processes with roles, in: A. Saffiotti (Ed.), IROS'02 Workshop on Cooperative Robotics, Lausanne, Switzerland, 2002, pp. 66–73.

[6] J.F. Nash, Equilibrium points in *n*-person games, in: Proceedings of the National Academy of Science, 1950, pp. 48–49.

[7] C. Boutilier, Planning, learning and coordination in multiagent decision processes, in: Proceedings of the Conference on Theoretical Aspects of Rationality and Knowledge, 1996, pp. 195–202.

[8] X. Wang, T. Sandholm, Reinforcement learning to play an optimal Nash equilibrium in team Markov games, in: Advances in Neural Information Processing Systems, vol. 15, MIT Press, Cambridge, MA, 2003.

[9] M. Bowling, M. Veloso, Multiagent learning using a variable learning rate, Artificial Intelligence 136 (8) (2002) 215–250.

[10] N.L. Zhang, D. Poole, Exploiting causal independence in bayesian network inference, Journal of Artificial Intelligence Research 5 (1996) 301–328.

[11] S. Arnborg, D. Corneil, A. Proskurowski, Complexity of finding embedding in a k-tree, SIAM J. Algebraic Discrete Meth. 8 (1987) 277–284.

[12] U. Bertelé, F. Brioschir, Nonserial Dynamic Programming, Academic Press, 1972.

[13] S. Kutten, Stepwise construction of an efficient distributed traversing algorithm for general strongly connected directed networks, in: Proceedings of the 9th International Conference on Computer Communications (CCC'88), Elsevier, Tel Aviv, Israel, 1988, pp. 446–452.

[14] I. Noda, H. Matsubara, K. Hiraki, I. Frank, Soccer Server: a tool for research on multi-agent systems, Applied Artificial Intelligence 12 (1998) 233–250.

[15] M. Chen, E. Foroughi, F. Heintz, Z. Huang, S. Kapetanakis, K. Kostiadis, J. Kummeneje, I. Noda, O. Obst, P. Riley, T. Steffens, Y. Wang, X. Yin, RoboCup Soccer Server for Soccer Server Version 7.07 and later, 2002. http://sserver.sourceforge.net/.

[16] L. Iocchi, D. Nardi, M. Piaggio, A. Sgorbissa, Distributed coordination in heterogeneous multi-robot systems, Autonomous Robots 15 (2) (2003) 155–168.

[17] P. Stone, M. Veloso, Task decomposition, dynamic role assignment and low-bandwidth communication for real-time strategic teamwork, Artificial Intelligence 110 (2) (1999) 241–273.

[18] M. Tambe, Towards flexible teamwork, Journal of Artificial Intelligence Research 7 (1997) 83–124.

[19] R. Fagin, J. Halpern, Y. Moses, M. Vardi, Reasoning About Knowledge, MIT Press, Cambridge, MA, 1995.

[20] R. de Boer, J.R. Kok, The incremental development of a synthetic multi-agent system: The UvA Trilearn 2001 Robotic Soccer Simulation Team, Master's Thesis, University of Amsterdam , The Netherlands, February 2002.

[21] P. Stone, D. McAllester, An architecture for action selection in robotic soccer, in: Proceedings of the Fifth International Conference on Autonomous Agents, ACM Press, 2001, pp. 316–323.

[22] J.R. Kok, R. de Boer, N. Vlassis, Towards an optimal scoring policy for simulated soccer agents, in: M. Gini, W. Shen, C. Torras, H. Yuasa (Eds.), Proceedings of the 7th International Conference on Intelligent Autonomous Systems, IOS Press, Marina del Rey, CA, 2002, pp. 195–198.

[23] I. Frank, K. Anaka-Ishii, K. Arai, H. Matsubara, The statistics proxy server, in: P. Stone, T. Balch, G. Kraetszchmar (Eds.), RoboCup-2000: Robot Soccer World Cup IV, Springer-Verlag, Berlin, 2001, pp. 303–308.

[24] H.J. Levesque, P.R. Cohen, J.T.H. Nunes, On acting together, in: Proceedings of the AAAI-90, 1990, pp. 94–99.

[25] P.R. Cohen, H.J. Levesque, Confirmations and joint action, in: Proceedings of the 12th International Joint Conference on Artificial Intelligence, Morgan Kaufman Publishers, Inc., San Mateo, CA, 1991, pp. 951–957.

[26] B.J. Grosz, S. Kraus, Collaborative plans for complex group action, Artificial Intelligence 86 (2) (1996) 269–357.