An approach to noncommunicative multiagent coordination in continuous domains*

Jelle R. Kok Matthijs T. J. Spaan Nikos Vlassis
Intelligent Autonomous Systems Group, Informatics Institute
Faculty of Science, University of Amsterdam, The Netherlands
{jellekok,mtjspaan,vlassis}@science.uva.nl

Abstract

Principled game-theoretic techniques exist for solving the problem of action coordination in a group of agents, however they typically suffer from an exponential blowup of the action space when many agents are involved. Coordination graphs (Guestrin et. al., 2002) offer tractable approximations via a context-specific decomposition into smaller coordination problems, and they are based on an iterative communication-based action selection procedure. We propose two extensions that apply when the agents are embedded in a continuous domain and/or communication is unavailable.

1 Introduction

Multiagent Systems (Weiss, 1999) is a relatively new field that has received considerable attention both in theory and applications. From an AI perspective, we can think of a multiagent system as a collection of agents that coexist in an environment, interact (explicitly or implicitly) with each other, and try to optimize a performance measure.

In this work we are interested in fully cooperative multiagent systems in which all agents share a common goal. A key aspect in such a system is the problem of *coordination*: how the individual agents can best choose their actions in order to successfully achieve a common goal (Boutilier, 1996).

Although in principle game theoretic techniques can be applied to solve the coordination problem (Osborne and Rubinstein, 1994), in practical situations involving many agents, even modeling an n-person game is intractable: the joint action space is exponentially large in the number of agents. However, one can often exploit the particular structure of a coordination problem in order to reduce its complexity.

A recent approach involves the use of a *coordination graph (CG)* (Guestrin et al., 2002a). This is a graph where each node represents an agent, and edges between nodes indicate that the corresponding agents have to coordinate their actions. In a context-specific CG (Guestrin et al., 2002b) the topology of the graph is dynamically updated based on the current context.

In this paper we extend a CG in two ways. First, we focus on agents that are embedded in a continuous domain (for example robotic agents in a soccer field) and are able to perceive their surroundings with sensors. For such a multiagent system, connectivity relationships between nodes in the coordination graph imply spatial relationships between agents, while the context is characterized by a continuous state variable. We propose a way to 'discretize' the context by appropriately assigning *roles* to the agents (Spaan et al., 2002) and then coordinating the different roles.

A second extension involves the way the agents compute their joint action. In the original formulation of CG, an agent needs to exchange information to and from its neighbors in order to compute its optimal action. This requires a communication channel which can be sometimes either unavailable or very costly to use. We propose a modification to the variable elimination algorithm of (Guestrin et al., 2002a) that allows each agent to efficiently predict the optimal action of its neighboring agents, making communication unnecessary.

The setup of the paper is as follows. In Section 2 we review the coordination problem, and

* Proc. Benelearn'02, Annual Machine Learning Conf. of Belgium and The Netherlands, Utrecht, The Netherlands, December 2002.

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

Figure 1: A coordination game.

in Section 3 we explain the concept of a CG. In Section 4 we describe our extensions, the role-dependent context and the noncommunicative case. In Section 5 we show some examples and in Section 6 we conclude and give hints for further research.

2 The coordination problem

We review here the agent coordination problem from a game theoretic point of view. A strategic game (Osborne and Rubinstein, 1994) is a tuple $(n, A_{1..n}, R_{1..n})$ where n is the number of agents, A_i is the set of actions of agent i and R_i is the payoff function for agent i. This payoff function maps the selected joint action $A = A_1 \times ... \times A_n$ to a real value: $R_i(A) \rightarrow \mathbb{R}$. 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.

A fully cooperative setting corresponds to a so-called coordination game in which all agents share the same payoff function $R_1 = \ldots = R_n = R$. Figure 1 shows an example of a coordination game between two agents. Each agent can choose between two types of movies, either a thriller or a comedy. They do not know in advance which movie the other agent will choose. Choosing the same movie results in an optimal joint action which offers them payoff 1, otherwise they receive payoff 0. It is clear that the agents have to coordinate their actions to maximize their payoff.

Formally, the coordination problem can be seen as the problem of selecting one out of many Nash equilibria in a coordination game. A Nash equilibrium defines a joint action $a^* \in A$ with the property that for every agent i holds $R_i(a_i^*, a_{-i}^*) \geq R_i(a_i, a_{-i}^*)$ for all $a_i \in A_i$, 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 Figure 1 has two Nash equilibria corresponding to the situations where both agents select the same action.

There are several ways to solve a coordination game (Boutilier, 1996), for example by using communication or by imposing social conventions. The latter are constraints on the possible action choices of the agents. If we assume that the agents have the ability to identify one another, we can create a simple lexicographic convention using the following three assumptions:

- The set of agents is ordered.
- The set of actions of each agent is ordered.
- These orderings are common knowledge among agents (Geanakoplos, 1992).

The choice for an optimal joint action proceeds as follows. The first agent in the agent ordering chooses an optimal action (that corresponds to a Nash equilibrium) that appears first in its action ordering. The next agent then chooses its first optimal action in its action ordering given the first agent's choice. This procedure continues until all agents have chosen their actions. 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. If we would impose the ordering '$1 \succ 2$' (meaning that agent 1 has priority over agent 2) and 'thriller $\succ$ comedy' in our example, the second agent knows from the social conventions that the first will select the thriller and will therefore also choose the thriller.

In the above cases it is assumed that the Nash equilibria can be found and then coordination is the problem of selecting 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 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

A coordination graph (CG) represents the coordination requirements of a system (Guestrin et

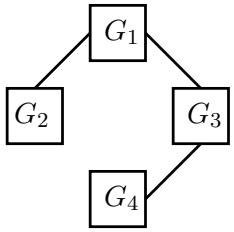


Figure 2: A CG for a 4-agent problem.

al., 2002a). A node in the graph represent an agent, while edges in the graph define dependencies between agents. Only agents that are interconnected have to coordinate their actions at any particular instant. Figure 2 shows a possible CG for a 4-agent problem. In this example, G_2 has to coordinate with G_1, G_4 has to coordinate with G_3, G_3 has to coordinate with both G_4 and G_1, and G_1 has to coordinate with both G_2 and G_3. Using such a graph, the global coordination problem can be replaced by a number of easier local coordination problems.

If the global payoff function can be decomposed as a sum of individual payoff functions, then solving for the joint optimal action can be done efficiently using a variable elimination algorithm (Guestrin et al., 2002a). The algorithm assumes an a priori elimination order that is common knowledge among the agents, and that each agent knows its neighbors in the graph (but not necessarily their payoff function which might depend on other agents). Each agent is 'eliminated' from the graph by solving a local optimization problem that involves only this agent and its neighbors: the agent collects from its neighbors all relevant payoff functions, then optimizes its decision conditionally on its neighbors' decisions, and communicates the resulting 'conditional' payoff function back to its neighbors. A next agent is selected from the list and the process continues. When all agents have been eliminated, each agent communicates its decision to its neighbors in the reverse elimination order.

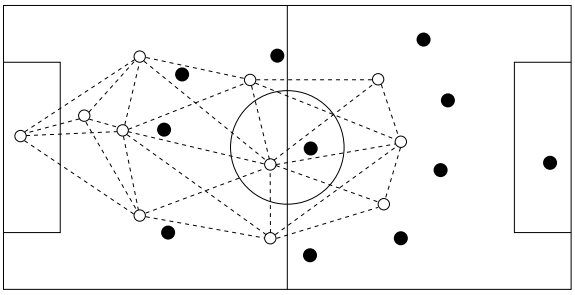
The local payoff functions can be matrix-based (Guestrin et al., 2002a) or rule-based (Guestrin et al., 2002b). In the latter case it is possible to use context-specific information to dynamically update the graph topology. A 'value rule' specifies how an agent's payoff depends on the current context, the latter being defined as a propositional rule over the state variables and the actions of the agent's neighbors. By conditioning on the current state the agents can discard all irrelevant rules, and this way the CG can be dynamically updated and simplified. Consider for example the situation where two plumbers have to fix the drainage system in a house. A value rule can specify that when the two plumbers are working in the same house they will get in each other's way, in which case the total payoff is decreased. In case the two plumbers are working in different houses, this value rule will not apply and the dependency in the graph is dynamically removed.

A limitation of this approach is that it is based on propositional rules and therefore only applies to discrete domains. Furthermore, in the variable elimination algorithm all coordinating agents must explicitly communicate their local payoff functions and their chosen actions using a message passing scheme. In the following we show how we can obtain context-specificity in a coordination graph when the agents reside in a continuous domain, and show how it is possible for each agent to predict the selected actions of its neighbors when communication is unavailable.

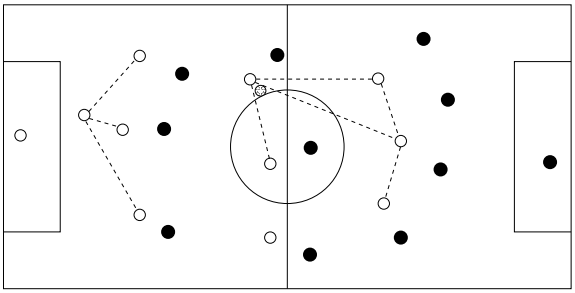
4 Coordination graphs in continuous domains

We are interested in problems where the agents 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 (see (de Boer and Kok, 2002) and references therein) in which a team of eleven agents have to fulfill a common goal (scoring more goals than your opponent). Depending on the situation, certain agents on the field have to coordinate their actions, for example the agent that controls the ball must decide to which nearby agent to pass, etc. 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.

We show an example of a continuous-domain



(a) Coordination graph.



(b) Reduced graph.

Figure 3: A coordination graph (a), and its context-specific reduction (b).

CG using the soccer domain. Figure 3(a) shows a picture of an a priori defined full coordination graph in which the dependencies between the teammates (represented by the open circles) are displayed. Based on the current context, e.g., the position of the ball in the field, the graph can be reduced as shown in Figure 3(b). The subgraph located in the left side of the field represents the relationship between the defenders trying to keep up a well-balanced defense. The subgraph on the right illustrates the local coordination game of the agent controlling the ball and the potential pass receivers. During the game, the coordination graph is continuously updated to reflect the current situation on the field.

Each node in such a CG has a natural 'location' within the domain while coordination dependencies automatically imply spatial relationships among agents. Moreover, contrary to the rule-based approach of (Guestrin et al., 2002b), the graph topology must depend on a context that is defined over a continuous state variable. In the above example, the context is based on the position of the ball which is a real variable having the soccer field as domain. We elaborate on this issue next.

4.1 Context-specificity based on 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 (Spaan et al., 2002). 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.

Given a particular local situation, each agent is assigned a role that is computed based on a role assignment function that is common knowledge among agents. The set of roles is finite and ordered, so 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. Environment-dependent 'potential' functions can be used to determine how appropriate an agent is for a particular role given the current context. For details on the assignment of roles to agents see (Spaan et al., 2002).

Such an assignment of roles provides a natural way to parametrize 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. For this, a priori rules exist that specify which roles should be coordinated and how.

As an example, consider again the left subgraph in Figure 3(b) involving four agents that organize the defense. The leftmost agent takes the role of sweeper while the other three all take the role of defender. It is common knowledge among the agents that the sweeper has to cover the space between the defenders and the goalkeeper to allow the defenders to advance to support the attack. As long as the four agents

agree on their role assignment the problem of their coordination is simplified: the defenders only need to take into account the action of the sweeper in their strategy (apart from other factors such as the opponents) making sure it is the most retracted field player. In their local coordination game they do not need to consider other teammates such the goalkeeper or the attackers. Several other local coordination games could be going on at the same time (e.g., in the attack) without interfering with each other.

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. A particular assignment of k roles to a group of agents with the roles ordered according to their importance, can be regarded as instantiation of a discrete context variable that can take $O(k!)$ possible values, corresponding to all possible assignments of the roles to agents.

In practice, a simple hierarchical role assignment scheme can be used, for example the two roles 'active' and 'passive' can be first assigned based on who is in control of the ball, then among all 'passive' agents additional roles can be assigned like 'sweeper' or 'striker', etc. In other cases, a particular context may reduce the number of required roles to a manageable quantity. In soccer, for example, k is often equal to 2, which resembles the situation where a player needs to pass the ball to another player (see also Section 5).

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 only one Nash equilibrium. For example, in Figure 1, if agent 2 is assigned a role that forbids him to select the action 'thriller' (e.g., because he is under 16), then agent 1, assuming he knows the role of agent 2, can safely choose 'comedy' resulting in coordination. Note there is only one Nash equilibrium in the subgame formed by removing the action 'thriller' from the action set of agent 2.

4.2 Non-communicating agents

Variable elimination in a CG requires that each agent first receives the payoff functions of its neighboring agents, and after computing its optimal conditional strategy it 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. The elimination order is a priori defined and is common knowledge among the agents.

When communication is unavailable the variable elimination algorithm can still be used 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 second requirement in fact frees agents from having to communicate their local payoff functions during optimization.

Moreover, in the noncommunicative case the elimination order neither has to be fixed in advance nor has to be common knowledge among all agents as in (Guestrin et al., 2002a), but each agent is free to choose *any* elimination order, e.g., 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.

In summary, each agent i maintains a pool of payoff functions, corresponding to all payoff functions of the agents in its subgraph. Starting from itself, agent i keeps eliminating agents using an appropriate elimination order, until it computes its own optimal action unconditionally on the actions of the others. For each eliminated agent j, the newly generated payoff functions are introduced into the pool of payoff functions of agent i and the process continues. In the worst case, agent i needs to eliminate all agents $j \neq i$, for j reachable from i. Note that, although each agent computes its own action in a different way (during optimization the pool will look different for different agents), the resulting joint action will always be the optimal one.

In terms of complexity, the computational costs for each individual agent are clearly increased to compensate for the unavailable com-

munication. 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 of the complete algorithm 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.

Finally, we note that the common knowledge assumption is strong and even in cases where communication is available it cannot always be guaranteed (Fagin et al., 1995). 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 the left subgraph of Figure 3(b) the particular role assignment may require that all four 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., ensuring 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, as explained above.

5 Experiments

We have applied the above ideas in our simulation robot soccer team (de Boer and Kok, 2002) with promising results. In the current phase we have not developed the CG framework to its full extent, but have tested it on simple situations where useful intuition can be gained.

We have implemented a simple role assignment function that assigns the role 'active' or 'passive' to a teammate based on whether it has the ball or not (for simplicity we focus here on the case where our team has the ball). At any instant only one agent is active and all the other 10 teammates are passive. Such a situation is shown in Figure 4 where one active and three passive teammates have to pairwise coordinate their actions.

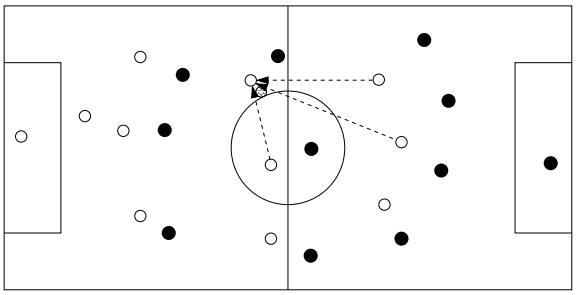


Figure 4: A simple situation involving one active and three passive agents.

Moreover, by construction an agent in a passive role always performs the same action, namely, moving towards its strategic position. The latter is computed based on the agent's home position (which is fixed throughout the game and known to all agents) and the position of the ball in the field which serves as an attraction point. As mentioned in section 4.1, such a drastic reduction of an action set greatly simplifies the local coordination game, because now the action choices of the three passive agents do not depend on the action choice of the active agent. In Figure 4 this is depicted by the directed edges between the agents.

Assuming the assignment of roles to the agents is common knowledge among reachable agents, the coordination problem resides now fully by the active agent. The latter has to choose one of the three teammates to pass the ball to, while we have assumed that the teammates follow their strategy independently of what the active or other passive agents do. Moreover, assuming that the active agent can also observe the position of the ball, it can predict the strategic position and thus the optimal action of each passive agent. The active player can now select to pass to the teammate that results in the highest future reward for the local coordination game; it will pass to the predicted position of the teammate with the maximum clearance from the opponents. Since the simula-

	With CG	Without
Wins	7	1
Draws	2	2
Losses	1	7
Avg. score	2.1	0.9
St. dev.	1.05	0.9

Table 1: Results of 10 games against ourselves, with and without CG.

tion server dynamics are known, predicting the one-step look-ahead reward is trivial (de Boer and Kok, 2002).

To test this approach we played games against ourselves, with one team using a CG and one team using no coordination at all during passing. In the latter case an active player would simply pass the ball to the last observed position of its teammate. Table 1 shows the results over the course of 10 full-length games. The results show that even the use of such a limited-scope CG has a positive effect on the performance on the team as a whole. Moreover, it turned out that the only statistically significant difference between the two teams was in passing. The successful passing percentage over these 10 matches was 80.12% for the team with the CG and 72.56% for the team without. These percentages indicate that due to the better coordination of the teammates, fewer mistakes were made when the ball was passed from one teammate to the other.

6 Conclusions and future work

We proposed two extensions to the framework of coordination graphs (Guestrin et al., 2002a) for the cases where the agents are embedded in a continuous domain and/or communication is unavailable. We argued that context-specificity is possible by appropriately assigning roles to the agents given a local situation. We also showed that we can dispense with communication if additional assumptions about common knowledge are introduced. We have not fully exploited the proposed framework in practice, but preliminary experiments in simulated soccer give promising results.

As future work, we first want to investigate the connotations of the common knowledge assumptions and how such knowledge can be obtained in practical situations. Second, we are interested in applying reinforcement learning techniques to a continuous-domain CG in order to learn the payoff functions in an automatic way, and we are looking for ways to efficiently plan ahead in a CG when an environment model is available. Finally, from an application point of view we want to apply the CG model to its full extent to the simulation RoboCup, where the agents need to continuously coordinate their actions, the context is time- and space-varying, and communication is restricted.

Acknowledgements

We thank the reviewers for their motivating comments. 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

C. Boutilier. 1996. Planning, learning and coordination in multiagent decision processes. In *Proc. Conf. on Theoretical Aspects of Rationality and Knowledge*.

R. de Boer and J. R. Kok. 2002. 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.

R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. 1995. *Reasoning about Knowledge*. The MIT Press, Cambridge, MA.

J. Geanakoplos. 1992. Common knowledge. *J. of Economic Perspectives*, 6(4):53–82.

C. Guestrin, D. Koller, and R. Parr. 2002a. Multiagent planning with factored MDPs. In *Advances in Neural Information Processing Systems 14*. The MIT Press.

C. Guestrin, S. Venkataraman, and D. Koller. 2002b. Context-specific multiagent coordination and planning with factored MDPs. In *AAAI 8th Nation. Conf. on Artificial Intelligence*, Edmonton, Canada, July.

M. J. Osborne and A. Rubinstein. 1994. *A course in game theory*. MIT Press.

M. T. J. Spaan, N. Vlassis, and F. C. A. Groen. 2002. High level coordination of agents based on multiagent Markov decision processes with roles. In A. Saffiotti, editor, *IROS'02 Workshop on Cooperative Robotics*, Lausanne, Switzerland, October.

G. Weiss, editor. 1999. *Multiagent Systems: a Modern Approach to Distributed Artificial Intelligence*. MIT Press.