Anytime algorithms for multiagent decision making using coordination graphs*

N. Vlassis R. Elhorst J. R. Kok
Informatics Institute, University of Amsterdam, The Netherlands
{vlassis,reinhrst,jellekok}@science.uva.nl

Abstract – *Coordination graphs provide a tractable framework for cooperative multiagent decision making by decomposing the global payoff function into a sum of local terms. In this paper we review some distributed algorithms for action selection in a coordination graph and discuss their pros and cons. For real-time decision making we emphasize the need for anytime algorithms for action selection: these are algorithms that improve the quality of the solution over time. We describe variable elimination, coordinate ascent, and the max-plus algorithm, the latter being an instance of the belief propagation algorithm in Bayesian networks. We discuss some interesting open problems related to the use of the max-plus algorithm in real-time multiagent decision making.*

Keywords: Multiagent systems, real-time systems, decision making, coordination graph, variable elimination, coordinate ascent, max-plus algorithm.

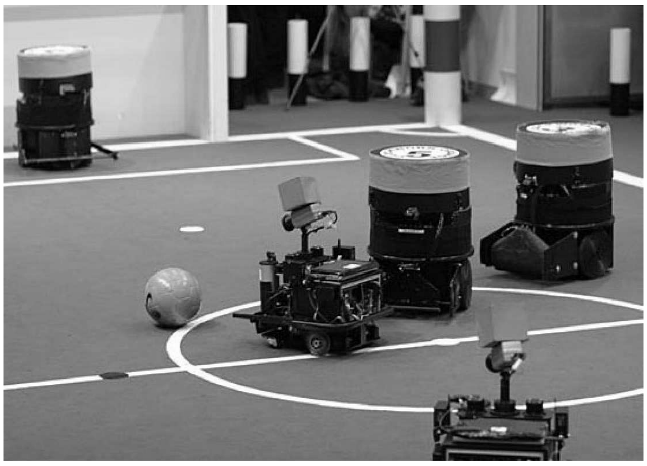
Figure 1: A robot soccer team is an example of a real-time cooperative multiagent system.

1 Introduction

Multiagent Systems (MAS) is an exciting new field with many theoretical and practical challenges [11, 8]. A MAS consists of a group of rational agents that can potentially interact with each other. These agents may have identical interests (like a team of soccer-playing robots, Fig. 1), conflicting interests (like two poker playing programs), or more general interests (like in e-commerce). A fundamental issue in MAS is how to implement agent-centric behavior that brings about desired system-wide behavior.

In this paper we are interested in *team* MAS, or fully cooperative multiagent systems where all agents share a common goal. A key aspect in such a system is the problem of *coordination*: how to ensure that the local (individual) decision making of each agent can produce globally good solutions for the team. One could use a centralized agent to solve the coordination problem: this agent could decide what action each individual agent should take, and then communicate these choices to each agent. However, such a system is not robust since a malfunction of the centralized agent could compromise the performance of the whole team.

Instead, in a team MAS we would like to have *decentralized* coordination: the agents should decide what actions to take by using a distributed protocol. In particular, when real-time decision making is in order, we would like such a protocol to be fast, robust to communication failures, and *anytime*. The latter suggests that we would like the quality of the solution to improve over time, and the agents should be able to report at any time the best solution (joint action) they have found so far. After some finite time we would like our protocol to converge to the optimal solution.

In this paper we first review in Section 2 the framework of coordination graphs (CG) for multiagent coordination, which allows for tractable representations when the number of agents is large [1]. Then we outline three classes of distributed algorithms for action selection in a CG, and focus on their real-time performance. We first discuss variable elimination, an exact method for action selection in CGs, and argue that this method may be inappropriate for real-time systems. In Section 3 we discuss two anytime algorithms for multiagent coordination: a coordinate ascent algorithm where the agents compute their individual actions in turn, and a max-plus algorithm in which the agents exchange appropriate payoff messages until a desired solution is computed. In Section 4 we summarize all three methods and discuss open issues for research.

*0-7803-8566-7/04/$20.00 © 2004 IEEE.

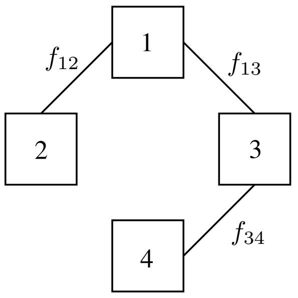

Figure 2: A coordination graph for a 4-agents problem.

2 Coordination graphs and variable elimination

We adopt a decision-theoretic approach to the coordination problem of n agents. In each time step we assume that each agent i chooses its individual action a_i from a set A_i, and the selected *joint* action $a = (a_1, \ldots, a_n)$ induces payoff to the team $u(a)$. The coordination problem is to find the optimal joint action a^* that maximizes $u(a)$, i.e., $a^* = \arg\max_a u(a)$. An obvious approach would involve enumerating all possible joint actions and selecting the one that maximizes $u(a)$, but this is clearly impractical: the joint action space $\times_i A_i$ is exponentially large in the number of agents n. A very large memory would be needed just to store the payoffs, apart from the cost of actually computing the optimal action.

It turns out that in many practical problems a complete enumeration of all joint actions is unnecessary: the payoff matrix $u(a)$ is sparse. This insight is exploited in the framework of *coordination graphs* (CG) [1]. A CG is a graph $G = (V, E)$ where each node in V represents an agent, and each edge in E defines a coordination dependency between two agents. An example graph with $n = 4$ agents is shown in Fig. 2.

The particular structure of a CG induces a decomposition of the global payoff function $u(a)$ into a linear combination of local payoff functions, each involving only few agents. For instance, in the graph of Fig. 2 the payoff function can be written:

$$u(a) = f_{12}(a_1, a_2) + f_{13}(a_1, a_3) + f_{34}(a_3, a_4). \tag{1}$$

Here, f_{13} for instance involves only agents 1 and 3, and for each pair of actions (a_1, a_3) contributes to the team local payoff $f_{13}(a_1, a_3)$.

In [1] an exact algorithm was proposed for finding the optimal joint action $a^* = \arg\max_a u(a)$ in a CG. The algorithm, called *variable elimination* (VE), is an iterative maximization procedure in which agents are eliminated one after the other from the graph.

We will illustrate VE on the above example. We start by eliminating agent 1 in (1). We collect all local payoff functions that involve agent 1, these are f_{12} and f_{13}. The maximum of $u(a)$ can then be written

$$\max_a u(a) = \max_{a_2,a_3,a_4} \Big\{ f_{34}(a_3, a_4) + \max_{a_1} \big[f_{12}(a_1, a_2) + f_{13}(a_1, a_3) \big] \Big\}. \tag{2}$$

Next we perform the inner maximization over the actions of agent 1. For each combination of actions of agents 2 and 3, agent 1 must choose an action that maximizes $f_{12} + f_{13}$. This results in a best-response function (conditional strategy) $B_1(a_2, a_3)$ for agent 1, given the actions of agents 2 and 3. The above maximization and the computation of the best-response function of agent 1 define a new payoff function $\phi_{23}(a_2, a_3) = \max_{a_1}[f_{12}(a_1, a_2) + f_{13}(a_1, a_3)]$ that is independent of a_1. Agent 1 has been eliminated. The maximum (2) becomes

$$\max_a u(a) = \max_{a_2,a_3,a_4} \big[f_{34}(a_3, a_4) + \phi_{23}(a_2, a_3) \big]. \tag{3}$$

We can now eliminate agent 2 as we did with agent 1. In (3), only ϕ_{23} involves a_2, and maximization of ϕ_{23} over a_2 gives the best-response function $B_2(a_3)$ of agent 2 which is a function of a_3 only. This in turn defines a new payoff function $\phi_3(a_3)$, and agent 2 is eliminated. Now we can write

$$\max_a u(a) = \max_{a_3,a_4} \big[f_{34}(a_3, a_4) + \phi_3(a_3) \big]. \tag{4}$$

Agent 3 is eliminated next, resulting in $B_3(a_4)$ and a new payoff function $\phi_4(a_4)$. Finally, $\max_a u(a) = \max_{a_4} \phi_4(a_4)$, and since all other agents have been eliminated, agent 4 can simply choose an action a_4^* that maximizes $\phi_4(a_4)$.

The above procedure computes an optimal action only for the last eliminated agent (assuming that the graph is connected). For the other agents it computes only conditional strategies. A second pass in the reverse elimination order is needed so that all agents compute their optimal (unconditional) actions from their best-response functions. Thus, in the above example, plugging a_4^* into $B_3(a_4)$ gives the optimal action a_3^* of agent 3. Similarly, we get a_2^* from $B_2(a_3^*)$ and a_1^* from $B_1(a_2^*, a_3^*)$, and thus we have computed the joint optimal action $a^* = (a_1^*, a_2^*, a_3^*, a_4^*)$. Note that one agent may have more than one best-response actions, in which case it can arbitrarily choose one of them (and then communicate it to each agent that needs it).

There are two limitations of VE that we are addressing here. First, the algorithm can be slow in certain cases, as it is exponential in the induced width of the graph (the size of the largest clique computed during node elimination). In the worst case VE scales exponentially in n. Second, VE may not always be appropriate for real-time multiagent systems where decision making must often be done under time constraints: typically, there is a deadline after which the payoff of the agents becomes zero. One example is robot soccer, where each agent has a relatively small amount of time available for deliberation. In these cases, an anytime algorithm

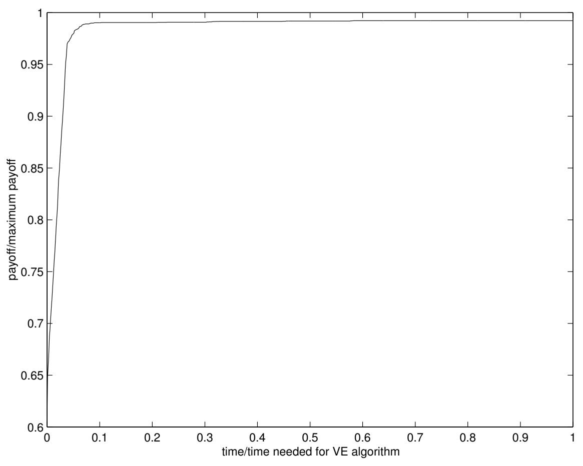

Figure 3: CA vs. VE: 800 loosely connected agents.

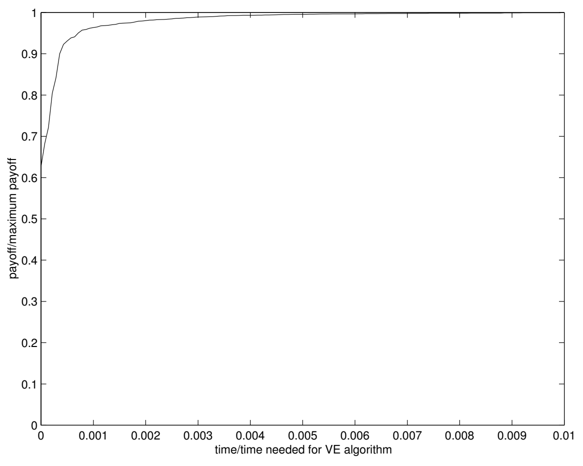

Figure 4: CA vs. VE: 13 densely connected agents.

would be more appropriate, one that improves the quality of the solution over time and eventually (given sufficient time) computes the optimal solution.

3 Anytime algorithms for action selection in coordination graphs

We describe here two classes of anytime algorithms for action selection in CGs, as alternatives to variable elimination, that are more appropriate for real-time systems.

3.1 Coordinate ascent

A simple anytime algorithm for action selection is a *coordinate ascent* (CA) with random restarts. Initially, each agent chooses its individual action, for instance randomly or using some local heuristics, resulting in a joint action $a^{(0)}$. At time step t, all agents fix their actions except for (a randomly selected) agent i. Using (1), this agent computes its conditional payoff function $u(a_i|a_{-i}^{(t)})$, where $a_{-i}^{(t)}$ refers to the vector of fixed actions of all agents except agent i. Then agent i maximizes $u(a_i|a_{-i}^{(t)})$ over its individual actions a_i, producing $a_i^{(t+1)}$. This action then replaces $a_i^{(t)}$ in $a^{(t)}$ to give $a^{(t+1)}$, another agent is selected, and so on, until $u(a^{(t+\tau)})$ does not improve anymore. The latter is a local maximum of the payoff function $u(a)$. If more time is available, another starting configuration $a^{(0)}$ is randomly selected and the above procedure is repeated. When the deadline expires, the joint action with the highest payoff is reported.

Note that, by construction, the payoff function $u(a)$ increases in each step of the algorithm, and therefore the resulting algorithm is anytime. Moreover, the graphical representation of (1) suggests a message passing scheme for action updating: after an agent computes a new individual action, it communicates this information only to its immediate neighbors in the graph. This way, a global solution can be computed by only local interactions. CA has been used in [2], in a problem involving global payoff functions with local constraints.

The CA algorithm constitutes an efficient, anytime algorithm that admits a distributed implementation. In many practical problems it can compute the optimal solution (or get very close to it) in a fraction of the time that VE takes. In Fig. 3-4 we show some results comparing CA with VE in randomly generated graphs. In both plots we show the maximum payoff computed by CA ($\%$ fraction relative to VE) vs. the total runtime of CA ($\%$ fraction relative to VE). In most cases CA computes solutions close to optimal in a fraction of the time that VE needs.

However, in general it is difficult to provide guarantees on the behavior of CA. Depending on the shape of the function $u(a)$, CA may need many random restarts to reach the global maximum of $u(a)$. A more sophisticated approach would be to search for the optimal joint action using a population of candidate configurations, properly selected in each optimization step via evolutionary techniques [6].

3.2 The max-plus algorithm

The *max-plus* algorithm is analogous to the sum-product or belief propagation algorithm used for inference in graphical models [7, 5, 9, 10]. It is easy to see that action selection in a CG is equivalent to computing the *maximum a posteriori* (MAP) configuration in an (unnormalized) undirected graphical model defined through a set of potential functions as in (1). In the max-plus algorithm—when viewed in the context of multiagent coordination—the agents exchange messages with each other, where each message can be regarded as a local payoff function. A nice property of max-plus is that upon convergence, and depending on the structure of the graph, the optimal joint action can be computed by only local computations. In the sequel we follow [9], translating their results into our multiagent decision making problem.

Suppose that we have n agents, and a coordination graph $G = (V, E)$ with V vertices and E edges that defines a payoff function as a sum of 2-agent local payoffs:

$$u(a) = \sum_{(i,j)\in E} f_{ij}(a_i, a_j). \tag{5}$$

Here (i,j) denotes a pair of neighboring agents (an edge in G), and f_{ij} is a local payoff function that maps a pair of actions (a_i, a_j) to a real number $f_{ij}(a_i, a_j)$. In each time step, each agent i (node in G) sends a message μ_{ij} to each of its neighbors $j \in \Gamma(i)$, where μ_{ij} is a (local) payoff function that maps an action a_j of agent j to a real number $\mu_{ij}(a_j)$. We further define:

$$g_i = \sum_{j \in \Gamma(i)} \mu_{ji},$$
$$g_{ij} = f_{ij} + \sum_{k \in \Gamma(i) \setminus j} \mu_{ki} + \sum_{k \in \Gamma(j) \setminus i} \mu_{kj},$$

where the notation $\Gamma(i) \setminus j$ means all neighbors of node i except node j. Clearly, g_i is a 1-agent payoff function and g_{ij} is a 2-agent payoff function.

Then we can easily show (by direct substitution) that if we have reached a 'fixed point' where the communicated messages among the agents do not change anymore, then the set of local payoff functions g_i, g_{ij} define a reparametrization of the original payoff function:

$$u(a) = \sum_{i \in V} g_i + \sum_{(i,j) \in E} (g_{ij} - g_i - g_j). \qquad (6)$$

Moreover, suppose that this fixed point has been reached by messages defined as follows:

$$\mu_{ij}(a_j) = \max_{a_i} \big\{ f_{ij}(a_i, a_j) + \sum_{k \in \Gamma(i) \setminus j} \mu_{ki}(a_i) \big\}. \qquad (7)$$

Then, we can easily verify that the following consistency property holds:

$$g_i(a_i) = \max_{a_j} g_{ij}(a_i, a_j) \qquad (8)$$

where j is an arbitrary neighbor of i.

From the above, the following important results follow [7, 9]. When the graph G is cycle-free (a tree), then max-plus always converges after a finite number of steps to a fixed point of the above message passing procedure. In this case, for the local functions g_i, g_{ij} holds:

$$g_i(a_i) = \max_{\{a' | a'_i = a_i\}} u(a'),$$
$$g_{ij}(a_i, a_j) = \max_{\{a' | (a'_i, a'_j) = (a_i, a_j)\}} u(a').$$

Consequently, if each individually (per agent) optimal action $a^*_i = \arg\max_{a_i} g_i(a_i)$ is unique for all i, then the globally optimal action $a^* = \arg\max_a u(a)$ is also unique and has elements $a^* = (a^*_i)$ computed by only local optimizations (each node maximizes $g_i(a_i)$ separately). If the local maximizers are not unique, an optimal joint action can still be computed by a straightforward dynamic programming technique [9, sec. 3.1].

The importance of the above result is that a difficult global optimization problem is transformed to a set of easy local optimization problems, one for each agent, using local message passing. Under the conditions stated above, this automatically defines an anytime algorithm: assuming that each agent can evaluate $u(a)$ for any a, the anytime solution is formed by the best (in terms of u) vector of local maximizers $a^*_i = \arg\max_{a_i} g_i(a_i)$ found so far. According to the above, after a finite number of steps we are guaranteed to find the optimal joint action.

In graphs with cycles, the above result does not hold anymore, and there are no guarantees that either max-plus will converge or that the local maximizers $a^*_i = \arg\max_{a_i} g_i(a_i)$ will correspond to the global optimum. In [9] it was shown that a fixed point of message passing exists in graphs with cycles, but there is no known algorithm yet that can provably converge to such a solution. Yet, bounds are available that characterize the quality of the max-plus solution if the algorithm converges [9].

4 Discussion and conclusions

We reviewed the framework of coordination graphs (CG) for multiagent coordination, and described the three existing algorithms for action selection in a CG, variable elimination, coordinate ascent, and the max-plus algorithm.

Variable elimination (VE) computes a solution by two passes over the graph. In the forward pass, agents are successively eliminated from the graph, until one agent is left for which decision making is easy. A second pass in the reverse elimination order is then employed to ensure that each agent computes its component of the optimal joint action. VE can be shown to always converge to the exact solution, independently of the elimination order and the structure of the graph, and it can be effective in loosely connected graphs. Its worst-case time complexity, however, is exponential in the number of agents involved in the graph, and therefore it can be slow in densely connected graphs. Moreover, VE is not appropriate for real-time systems as it requires that both passes terminate before a solution can be reported.

Coordinate ascent (CA) with random restarts is a very simple method in which each agent optimizes its own action only, given that the actions of all other agents remain fixed. This is repeated for all agents iteratively, until a local maximum of the global payoff function has been reached. A new initial configuration is then chosen, and the process is repeated. The method is very effective in practice, and it can be implemented in a distributed fashion [2]. CA will compute the optimal joint action in the limit of an infinite number of random restarts, but it is difficult to characterize its speed of convergence on arbitrary graphs.

Finally, the max-plus algorithm is similar to the belief propagation algorithm in Bayesian networks [7, 10]. It involves repeated passing of messages over the graph, each message being a local payoff function for the agent that receives the message. Due to its asynchronous nature, the algorithm is particularly appropriate for real-time multiagent systems. Moreover, strong theoretical results exist for the original algorithm and its variants, like global optimality in the case of cycle-free graphs and the existence of fixed points in arbitrary graphs. However, there is no message passing schedule yet that is provably convergent.

We see a few interesting open issues for further research, in particular related to the use of the max-plus algorithm. First, it would be useful to further characterize the anytime behavior of the algorithm, even in graphs without cycles. For instance, we would like to have a message passing schedule that ensures a monotone (and fast) increase of the global payoff value in each step. The results that we mentioned above guarantee that in cycle-free CGs, under mild conditions, max-plus will converge to the optimal joint action, but we would like to ensure high speed of convergence on the average.

A second issue is related to the convergence of max-plus on arbitrary graphs, which is an open problem. In recent work [10], a 'reweighted' version of max-plus has been proposed, that exhibits better convergence behavior than the original algorithm and for which stronger theoretical results can be formulated. It would be interesting to further investigate the applicability of these algorithms in the context of multiagent coordination.

Another line of research would be to use a message passing algorithm like max-plus for sequential decision making, like in Markov decision processes or reinforcement learning [1, 4]. Finally, from an application point of view, it would be interesting to test some of the above methods on large-scale problems like robot soccer [3].

Acknowledgments

We thank Carlos Guestrin and Taylan Cemgil for helpful discussions. 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] C. Guestrin, D. Koller, and R. Parr. Multiagent planning with factored MDPs. In *Advances in Neural Information Processing Systems 14*. The MIT Press, 2002.

[2] G. İnalhan, D. M. Stipanović, and C. J. Tomlin. Decentralized optimization with application to multiple aircraft coordination. In *Proc. IEEE Int. Conf. on Decision and Control*, Las Vegas, Nevada, 2002.

[3] J. R. Kok, M. T. J. Spaan, and N. Vlassis. Multi-robot decision making using coordination graphs. In *Proc. 11th Int. Conf. on Advanced Robotics*, Coimbra, Portugal, June 2003.

[4] J. R. Kok and N. Vlassis. Sparse cooperative Q-learning. In *Proc. 21st Int. Conf. on Machine Learning*, Banff, Canada, July 2004.

[5] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Trans. on Information Theory*, 47:498–519, 2001.

[6] H. Mühlenbein and T. Mahnig. FDA—a scalable evolutionary algorithm for the optimization of additively decomposed functions. *Evolutionary Computation*, 7:353–376, 1999.

[7] J. Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufman, San Mateo, 1988.

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

[9] M. Wainwright, T. Jaakkola, and A. Willsky. Tree consistency and bounds on the performance of the max-product algorithm and its generalizations. Technical report, P-2554, LIDS-MIT, 2002.

[10] M. Wainwright, T. Jaakkola, and A. Willsky. MAP estimation via agreement on (hyper)trees: Message-passing and linear programming approaches. Technical report, UCB/CSD-03-1269, UC Berkeley, 2003.

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