Multi-Objective Variable Elimination for Collaborative Graphical Games

(Extended Abstract)

Diederik M. Roijers
Informatics Institute
University of Amsterdam
Amsterdam, the Netherlands
d.m.roijers@uva.nl

Shimon Whiteson
Informatics Institute
University of Amsterdam
Amsterdam, the Netherlands
s.a.whiteson@uva.nl

Frans A. Oliehoek
Maastricht University
Maastricht, the Netherlands
frans.oliehoek@
maastrichtuniversity.nl

ABSTRACT

In this paper we propose *multi-objective variable elimination* (MOVE), an efficient solution method for *multi-objective collaborative graphical games* (MO-CoGGs), that exploits loose couplings. MOVE computes the convex coverage set, which can be much smaller than the Pareto front. In an empirical study, we show that MOVE can tackle multi-objective problems much faster than methods that do not exploit loose couplings.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: multi-agent systems

Keywords

Multiple objectives, game theory, coordination graphs

1. INTRODUCTION

In cooperative multi-agent systems, teams of agents must coordinate their behavior in order to maximize their common utility. Such systems are useful, not only for addressing tasks that are inherently distributed, but also for decomposing tasks that would otherwise be too complex to solve. Key to making coordination efficient is exploiting the fact that such tasks are typically *loosely coupled*, i.e., each agent's actions directly affect only a subset of the other agents. This independence can be captured in a graphical model and used to efficiently compute coordinated behavior [2, 3].

In this paper, we consider how to address cooperative multi-agent systems in which the agents have multiple objectives, i.e., the utility is vector-valued. Many real-world problems require incorporating multiple objectives. For instance, while recommending medical treatment, we need a system that maximizes the effectiveness of the treatment while minimizing the severity of the side effects [4].

If the vector-valued utility function can be *scalarized*, i.e., converted to a scalar function, then the original problem may be solvable with existing single-objective methods. In settings where this is impossible, e.g., because the weights are unknown at the time of planning we need methods that compute a set of solutions containing optimal solution for any weight setting.

We propose an efficient solution method for a multi-objective extension to a *collaborative graphical game* (also known as a *coordination graph* [2]). Our method, called *multi-objective variable elimination* (MOVE), extends *variable elimination* (VE) [2, 3]. The main idea is to replace the maximization in VE by an operator that prunes away dominated solutions (those not optimal for any weight setting).

In contrast to related methods for similar problems [5, 1], we prove correctness and give better complexity bounds. Furthermore, we focus on a stricter solution set. In the highly prevalent case that the scalarization is linear, this enables the algorithm to compute only the *convex cover* instead of the typically much larger *Pareto front*. We present an empirical study that shows that this method can tackle multi-objective problems much faster than those that do not exploit loose couplings or compute the Pareto front instead of the convex cover, and analyzes its scalability with respect to the number of objectives and the number of agents.

2. MODEL

We formalize the problem as a *multi-objective collaborative graphical game* (MO-CoGG), which is a tuple $\langle \mathcal{D}, \mathcal{A}, \mathcal{U} \rangle$. $\mathcal{D} = \{1, ..., n\}$ is the set of n agents. $\mathcal{A} = \mathcal{A}_i \times ... \times \mathcal{A}_n$ is the joint action space: the Cartesian product of the finite action spaces of all agents. A joint action is thus a tuple containing an action for each agent $\mathbf{a} = \langle a_1, ..., a_n \rangle$. $\mathcal{U} = \left\{\mathbf{u}^1, ..., \mathbf{u}^\rho\right\}$ is the set of ρ, d-dimensional *local payoff functions*, each $\mathbf{u}^e(\mathbf{a}_e)$ depends on a subset of agents. The total team payoff is the (vector) sum of local payoffs: $\mathbf{u}(\mathbf{a}) = \sum_{e=1}^{\rho} \mathbf{u}^e(\mathbf{a}_e)$. The game is collaborative because all agents share the payoff function $u(\mathbf{a})$. Where the local payoff functions are specified

We assume there exists a scalarization function that converts $\mathbf{u}(\mathbf{a})$ to a scalar payoff function $u_{\mathbf{w}}(\mathbf{a})$ parameterized by a weight vector $\mathbf{w}$, which is unknown *when the game is solved* but known when the agents must actually select actions. Consequently, we want to find the *coverage set (CS)*, i.e., all solutions $\mathbf{a}$ that are optimal for some $\mathbf{w}$:

$$\mathcal{A}^* = \left\{\mathbf{a} : \exists \mathbf{w} \forall \mathbf{a}' \; u_{\mathbf{w}}(\mathbf{a}) \geq u_{\mathbf{w}}(\mathbf{a}')\right\} \subseteq \mathcal{A}. \quad (1)$$

Solutions $\mathbf{a}$ that are not part of the CS (i.e., are not maximal for any weight vector $\mathbf{w}$), are *dominated*.

When the scalarized payoff is linear – a convex combination of individual objectives: $u_{\mathbf{w}}(\mathbf{a}) = \mathbf{w} \cdot \mathbf{u}(\mathbf{a})$, we refer to the CS given by (1) as the CS-C, which excludes all *C-*

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*, Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.

Algorithm 1: Agent elimination

$f^{new}(\mathbf{a}_{n_i}) \leftarrow$ a new factor
foreach $\mathbf{a}_{n_i} \in \mathcal{A}_{n_i}$ **do**
  $f^{new}(\mathbf{a}_{n_i}) \leftarrow \texttt{prune}\left(\bigcup_{a_i} \bigoplus_{f^e \in \mathcal{F}_i} f^e(\mathbf{a}_e)\right)$
end
$\mathcal{F} \leftarrow \mathcal{F} \setminus \mathcal{F}_i \cup \{f^{new}\}$

dominated joint actions. The CS-C is typically smaller than the Pareto front, which is another solution concept [5].

3. THE MOVE ALGORITHM

MOVE extends the VE algorithm, which operates by iteratively 'eliminating' agents. In order to eliminate an agent i, all the payoff functions in which it participates are gathered in a set $\mathcal{F}_i$, which is then used to define its local payoff: $f_i(\mathbf{a}_{n_i}, a_i) = \sum_{u^e \in \mathcal{F}_i} u^e(\mathbf{a}_e)$. This in turn is used to compute the agent's best-response value:

$$f(\mathbf{a}_{n_i}) = \max\left(\cup_{a_i} \{f_i(\mathbf{a}_{n_i}, a_i)\}\right) \quad (2)$$

Given this function f, we can remove all factors of $\mathcal{F}_i$ and agent i from the problem, and introduce f as a new local payoff function.

Like single-objective VE, MOVE eliminates agents in sequence. However, instead of computing a single optimal joint action, it computes the entire CS-C. Therefore the solution to a local subproblem is no longer a single action but a *local coverage set*. This implies that we first need replace the local payoff functions by *vector set factors* (VSFs), denoted f^e, that map a joint action of the agent in scope to a set of possible payoff vectors: $f^e(\mathbf{a}_e) \triangleq \{u^e(\mathbf{a}_e)\}$. Moreover, the max from (2) must be replaced with a pruning operator that calculates the CS-C. To do this, we define the 'local payoff set' $f_i(\mathbf{a}_{n_i}, a_i) = \oplus_{f^e \in \mathcal{F}_i} f^e(\mathbf{a}_e)$[1], and use it to define the multiobjective equivalent of best-response value:

$$f(\mathbf{a}_{n_i}) = \texttt{prune}\Big(\cup_{a_i} \{f_i(\mathbf{a}_{n_i}, a_i)\}\Big). \quad (3)$$

The pruning operator for calculating the CS-C uses linear programming, and requires $O(s_{in}s_p + s_pP(s_c))$ time, where s_{in}, s_p and s_c are the sizes of the input set, the Pareto front and the CS-C; P indicates the polynomial runtime of linear programming.

In MOVE, first all local payoff functions are translated to VSFs, then from the set of VSFs $\mathcal{F}$ we eliminate all agents in sequence, using Algorithm 1. It can be shown that this results in the CS-C when all agents are eliminated. MOVE runs in $O(n|\mathcal{A}_{max}|^w(s^*_{in}s^*_p + s^*_pP(s^*_c))$ time, where w is the induced width (like in VE), s^*_{in}, s^*_p and s^*_c are the maximal input, local Pareto front and local CS-C sizes during any agent elimination. By only pruning vectors that are Pareto dominated, a variant of MOVE ("Pareto Move") can compute the Pareto Front. To discriminate, we refer to the variant of MOVE introduced above as "Convex MOVE".

4. RESULTS

In order to show the scalability of Convex MOVE and compare it to a non-graphical approach (that loops over all joint actions) and Pareto MOVE, we test the algorithms on randomly generated MO-CoGGs and a benchmark we propose called Mining Day. In this benchmark, different amounts of gold and silver can be mined in mines that are geographically distributed by different worker groups, with a certain action radius.

[1] $A \oplus B = \{\mathbf{a} + \mathbf{b} : \mathbf{a} \in A \wedge \mathbf{b} \in B\}$ is the cross-sum.

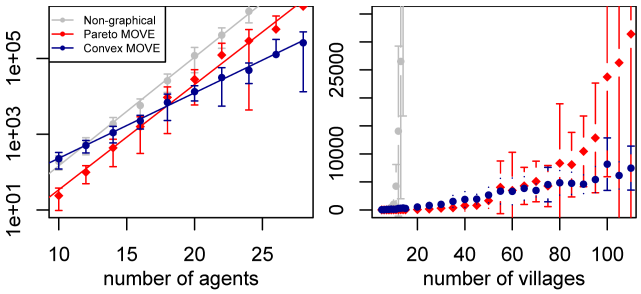

Figure 1: Runtimes (ms) for random graphs (left), and mining day (right)

The results for random graphs show that Convex MOVE outperforms the non-graphical approach when the number of agents increases (Figure 1 (left)). Generating MO-CoGGs with a number of agents n, a number of factors $\rho = 1.5n$ and 5 objectives, we show that for 12 agents Convex MOVE starts to outperform the non-graphical method. For 22 agents, MOVE is 17.6 times faster. Convex MOVE starts doing better than Pareto MOVE at 20 agents.

The most striking differences occurs when the problem is highly structured, as in *Mining Day*. While the runtime of a non-graphical approach increases exponentially in the number of agents, Pareto MOVE seems to be quadratic and Convex MOVE linear in number of agents (Figure 1 (right)). While the non-graphical approach cannot tackle problems when the number of agents is greater than 15 in less than 100s, Convex MOVE stays under ten seconds even at 100 agents.

5. CONCLUSIONS

We propose MOVE as a new method for multi-objective multi-agent graphical games. We show the correctness of MOVE and analyze its complexity. Our empirical study shows that MOVE can tackle multi-objective problems much faster than methods that do not exploit loose couplings. It also shows that for larger numbers of agents Convex MOVE is much faster than Pareto MOVE.

6. ACKNOWLEDGEMENTS

This research is supported by the NWO DTC-NCAP (#612.001.109) and NWO CATCH (#640.005.003) projects.

7. REFERENCES

[1] F. Delle Fave, R. Stranders, A. Rogers, and N. Jennings. Bounded decentralised coordination over multiple objectives. In *AAMAS*, pages 371–378, 2011.
[2] C. Guestrin, D. Koller, and R. Parr. Multiagent planning with factored MDPs. In *NIPS*, 2002.
[3] J. R. Kok and N. Vlassis. Collaborative multiagent reinforcement learning by payoff propagation. *J. Mach. Learn. Res.*, 7:1789–1828, Dec. 2006.
[4] D. J. Lizotte, M. Bowling, and S. A. Murphy. Efficient reinforcement learning with multiple reward functions for randomized clinical trial analysis. In *ICML*, 2010.
[5] E. Rollón and J. Larrosa. Bucket elimination for multiobjective optimization problems. *J Heuristics*, 12:307–328, 2006.