Distributed Search for Pure Nash Equilibria in Graphical Games

(Extended Abstract)

Omer Litov
Department of Computer Science
Ben-Gurion University of the Negev
Beer-Sheva, 84-105, Israel
litov@cs.bgu.ac.il

Amnon Meisels
Department of Computer Science
Ben-Gurion University of the Negev
Beer-Sheva, 84-105, Israel
am@cs.bgu.ac.il

ABSTRACT

Graphical games introduce a compact representation, where agents' outcomes depend only on their neighbors. A distributed search algorithm for pure Nash equilibria of graphical games is presented. The algorithm uses the analogy of graphical games with asymmetric distributed constraints optimization problems (ADCOPs). The proposed algorithm includes three components - an admissible pruning heuristic; a back-checking mechanism; and a pseudo tree representation of the game. An experimental evaluation of the components of the proposed search algorithm is presented for randomly generated networks of multiple agents. The major speedup over a naive search algorithm is shown to arise from the use of a pseudo tree representation. A simple assessment method of the privacy loss due to back-checking is presented and is shown to result in a tradeoff between the performance of the complete algorithm and its privacy loss.

Categories and Subject Descriptors

I.2.11 [**Distributed artificial intelligence**]: Multi-agent systems

General Terms

Algorithms

Keywords

Distributed problem solving; DCOP; Nash equilibrium; Privacy

1. INTRODUCTION

In many Multi Agent situations, agents interact with one another to achieve some goal. For cooperative agents, this goal may be a global objective such as the minimization of total cost in distributed constraints reasoning. For selfish agents, attempting to maximize their personal gains, Game Theory predicts that the outcome will be a stable state (e.g., an equilibrium) from which every agent will not wish to deviate.This notion of stability is a fundamental concept in game theory and has been a major focus of work in the field of Multi Agent Systems.

Graphical Games have been proposed as a compact representation of normal form games played over a graph [3]. It is a model that exploits the locality of interactions between agents and enables one to specify payoffs in terms of neighbors rather than in terms of the entire population of agents. Several algorithms have been proposed for finding an approximate **mixed** strategy equilibrium of graphical games, the most recent being ANT [2].

In a pure Nash equilibrium (PNE), which may not always exist, every agent chooses a single pure strategy. It is a clear solution concept for a multi-agent situation since it is deterministic in nature (unlike mixed Nash equilibria in which every agent is eventually required to draw its selected strategy from the probability distribution of its mixed strategy). Centralized solvers have been proposed (Gambit [4] and more recently ConGa [6]), although centralized algorithms may not be appropriate when privacy of gains is an issue, since it requires complete revelation of information from all of the agents.

The present paper focuses on finding Pure Nash Equilibria in a distributed environment and is based on modeling graphical games as (Asymmetric) Distributed Constraints Optimization Problems (DCOPs) [5]. Several tools for finding stable points are presented and are then combined into a complete algorithm. The model uses Asymmetric DCOPs (ADCOPs) [1], which are an extension of standard DCOPs. The proposed search algorithms for finding a PNE combines two well known DCOP techniques – a back checking procedure and a pseudo-tree.

An extensive experimental evaluation (abbreviately shown in figure 1) shows the improvement in performance of both run-time and network load, over a distributed version of the Gambit solver and over any of the algorithmic tools separately. An upper bound on the loss of privacy is calculated, to present the tradeoff between performance and privacy conservation. An improvement in performance of a factor 10 in speedup in exchange for an upper bound of 7% loss of privacy.

The next section describes the main concepts of the proposed algorithm.

2. FINDING A PURE NASH EQUILIBRIUM

Every graphical game with a finite set of strategies can be represented as an ADCOP in the following way: The set of players is represented by the set of agents $\mathcal{A}$, where each agent A_i has exactly one variable X_i. The set of strategies of A_i is represented by its domain D_i. The cost/payoff of every interaction between players in a given neighborhood (i.e., connected by edges in the graph of the game) is represented as an asymmetric constraint. Standard ADCOP algorithms search for the Social Welfare solution [1]. The present study focuses on search for pure Nash equilibria (PNEs), which will be addressed next.

Appears in: *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016), J. Thangarajah, K. Tuyls, C. Jonker, S. Marsella (eds.), May 9–13, 2016, Singapore.*

2.1 Back Checking algorithm

For simplicity, we start by describing a partial algorithm, which uses a back checking procedure, without the structure of a pseudo-tree. The proposed back checking algorithm is based on distributed backtracking search, and uses the following messages.

1. **CPA** - Current Partial Assignment.
2. **Backtrack**
3. **Check_PNE** - A request to check whether the value assignment $v_{i_k} \in CPA$ has the potential to be a best response.
4. **Response** - A response to a **Check_PNE** message.

The algorithm has each agent assign a value synchronously on the **CPA** message according to a predefined complete order of all the agents. After agent A_i assigns a value to its variable, and before it sends a **CPA** message to the next agent, it sends a **Check_PNE** message to every constrained agent before it. When an agent that is constrained with A_i receives a **Check_PNE** message it will check for the pruning criteria according to the heuristic defined below. Next, it will send back a **Response** message that states whether the current CPA can be continued. Only when agent A_i receives back all of the **Response** messages from its constrained predecessors it will proceed, either by pruning its value or by sending forward a **CPA**. When the last agent has received a positive **Response** message from all of its constrained ancestor a Pure Nash Equilibrium solution has been reached.

We must now define the method to identify obsolete value assignments, which can be pruned. We will define the following two heuristics.

$$LB(v_{i_k}) = CurrentCost(CPA, v_{i_k}) + \sum_{A_j \notin CPA} \min_{v_{j_q} \in D_j} Cost_i(v_{i_k}, v_{j_q}) \quad (1)$$

$$UB(v_{i_k}) = CurrentCost(CPA, v_{i_k}) + \sum_{A_j \notin CPA} \max_{v_{j_q} \in D_j} Cost_i(v_{i_k}, v_{j_q}) \quad (2)$$

where $Cost_i(v_{i_k}, v_{j_q})$ is defined to be the cost for agent A_i for the constraint between A_i's value v_{i_k} and A_j's value v_{j_q}, and $CurrentCost(CPA, v_{i_k}) = \sum_{v_j \in CPA} Cost_i(v_{i_k}, v_j)$.

It is possible to see that $LB(v_{i_k})$ (and $UB(v_{i_k})$) is a lower bound (and upper bound) on the cost of agent A_i for assigning v_{i_k}, respectively. It follows that if $\exists v_{i_q} \neq v_{i_k} : UB(v_{i_q}) < LB(v_{i_k})$, then the value v_{i_k} can be pruned from the search. This check is performed in two places: when an agent receives the **CPA** and attempt to assign a new value, and when an agent receives a **Check_PNE** (in order to inform the sending agent to prune its own value).

2.2 Using a Pseudo-Tree

The main component of the proposed search algorithm for a PNE uses a very well known tool for DCOP search algorithms – a Pseudo-Tree [5]. The pseudo-tree defines a partial order over the agents (unlike the total order used before), starting from the root down to the leafs. When an agent with more than one child needs to send forward a **CPA**, it sends it concurrently to every one of its children. Each leaf acts as a last agent in its branch and since there are multiple leaves the possible solutions must propagate up the tree to the root. This presents a problem for agents with more than one child. An agent may receive several **Check_PNE** messages from each sub-tree, and the solution may be any combination of the partial assignments of those messages.

In order to solve the problem described above one needs to define a new type of message – **Sub_Problem_PNE**. When a leaf agent receives a positive **Response** from all of its constrained agents it sends a **Sub_Problem_PNE** message to $Splitter(A_i)$. We define the $Splitter(A_i)$ agent to be the closest ancestor of agent A_i who has more than one child. $Branch(A_i)$ is defined to be the set of ancestors of agent A_i up to (and excluding) the $Splitter(A_i)$ agent. The **Sub_Problem_PNE** message contains a Partial Assignment PA. The assignment of every agent $A_j \in Branch(A_i)$ in PA is a best response. This is due to the fact that all agents sent a positive **Response** to the leaf for the given PA, in which all of their constrained agents are already assigned. The leaf agent continues the search, because it is not waiting for any response to that message. Agent A_i tries to combine the partial assignment from the **Sub_Problem_PNE** with every other partial assignment received from the other sub-trees, and check whether any one of the combinations is a possible PNE. For any Partial Assignment PA constructed from a combination of assignments received by **Sub_Problem_PNE** messages, if value assignment of A_i is a best response then PA may possibly be extended into a PNE for the entire problem and thus should be propagated up the tree. Agent A_i propagates every such PA by sending every agent $A_j \in Branch(A_i)$ a **Check_PNE** message and if all **Response**s are positive it then sends a **Sub_Problem_PNE** to $Splitter(i)$.

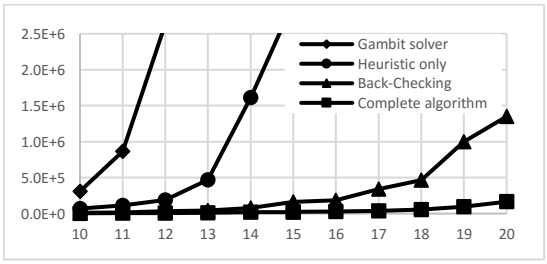

Figure 1: NCCC for variable number of agents

REFERENCES

[1] T. Grinshpoun, A. Grubshtein, R. Zivan, A. Netzer, and A. Meisels. Asymmetric distributed constraint optimization problems. *Journal of Artificial Intelligence Research*, pages 613–647, 2013.

[2] A. Grubshtein and A. Meisels. Finding a nash equilibrium by asynchronous backtracking. In *Principles and Practice of Constraint Programming - 18th International Conference, CP 2012, Québec City, QC, Canada, October 8-12, 2012. Proceedings*, pages 925–940, 2012.

[3] M. J. Kearns, M. L. Littman, and S. P. Singh. Graphical models for game theory. In *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, University of Washington, Seattle, Washington, USA, August 2-5, 2001*, pages 253–260, 2001.

[4] R. D. McKelvey, A. M. McLennan, and T. L. Turocy. Gambit: Software tools for game theory, version 14.1.0. `http://www.gambit-project.org`, 2014.

[5] P. J. Modi, W. Shen, M. Tambe, and M. Yokoo. ADOPT: asynchronous distributed constraints optimization with quality guarantees. *Artificial Intelligence*, 161:1-2:149–180, January 2005.

[6] T. Nguyen and A. Lallouet. A complete solver for constraint games. In *Principles and Practice of Constraint Programming - 20th International Conference, CP 2014, Lyon, France, September 8-12, 2014. Proceedings*, pages 58–74, 2014.