Heuristic Multiagent Planning with Self-Interested Agents (Extended Abstract)

Matt Crosby
School of Informatics
University of Edinburgh
Edinburgh EH8 9AB, UK
M.Crosby@ed.ac.uk

Michael Rovatsos
School of Informatics
University of Edinburgh
Edinburgh EH8 9AB, UK
M.Rovatsos@ed.ac.uk

ABSTRACT

The focus of multiagent planning research has recently turned towards domains with self-interested agents leading to the definition of Coalition–Planning Games (CoPGs). In this paper, we investigate algorithms for solving a restricted class of *"safe"* CoPGs, in which no agent can benefit from making another agent's plan invalid. We introduce a novel, generalised solution concept, and show how problems can be translated so that they can be solved by standard single–agent planners. However, standard planners cannot solve problems like this efficiently. We then introduce a new multiagent planning algorithm and the benefits of our approach are illustrated empirically in an example logistics domain.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Plan execution, formation, and generation, Heuristic Methods*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Algorithms, Performance, Experimentation, Theory

Keywords

Multiagent planning, single-agent planning, Coalition-Planning Games

1. INTRODUCTION

Historically, multiagent planning has assumed cooperative agents, and has mostly ignored issues associated with *strategic* behaviour among self-interested agents. Recently however, this problem has started to attract more interest [1, 5] and Brafman et al [2] have introduced Coalition-Planning Games (CoPGs), multiagent planning problems with self-interested but ready to cooperate agents.

We introduce a new solution concept for CoPGs that avoids some unintuitive properties of the concept existing in the literature and show that a restricted subset of CoPGs can be solved using existing planners. However, this approach fails to make use of the powerful heuristics tools provided by modern planners. We therefore introduce a novel algorithm that combines heuristic calculations with the distinction between an agent's public and internal actions as introduced in [1].

Cite as: Heuristic Multiagent Planning with Self-Interested Agents (Extended Abstract), Matt Crosby, Michael Rovatsos, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 1213-1214.

2. COPGS

Definition: A *Coalition-Planning Game* CoPG [2], with n agents $N = \{1, \ldots, n\}$, is an extension to a classical STRIPS planning problem and is represented by a 6-tuple $\Pi = \langle P, A, I, G, c, r \rangle$. P is a set of grounded atoms and $I \subseteq P$ represents the initial state of the world, $A = \{A_i\}_{i=1}^n$ comprises of a set of actions for each agent and $G = \{G_i\}_{i=1}^n$ contains a goal set for each agent, $c : A \rightarrow \Re^+$ is a cost function and $r : N \rightarrow \Re^+$ is a reward function. Agents are assumed to be self-interested, but able to form coalitions (costless binding agreements).

A solution to a planning problem Π is a *plan* $\pi = \{a_1, \ldots, a_n\}$, an ordered sequence of actions that can be executed in sequence.[1] The utility of an agent's plan is defined as:

$$u_i(\pi) = \begin{cases} r(i) - \sum_{a \in \{a \in \pi : a \in A_i\}} c(a) & \text{if } \pi \text{ achieves } G_i \\ - \sum_{a \in \{a \in \pi : a \in A_i\}} c(a) & \text{otherwise} \end{cases}$$

Let $u_S(\pi)$ represent the vector of utilities for agents in $S \subseteq N$. Let $(\pi_S, \pi_{S'})$ be the joint plan constructed by combining the plans π_S and $\pi_{S'}$ for disjoint subsets $S, S' \subseteq N$. We say a vector $u > u'$ if every element of u is greater than the equivalent element in u'. For $S \subseteq N$ we call $\Pi|_S$ the CoPG Π restricted to S defined as: $\Pi|_S = \langle P, \cup_{i \in S} A_i, I, \cup_{i \in S} G_i, c, r \rangle$ We use sol(Π) to represent the set of plans that are possible solutions to Π.

3. SOLUTION CONCEPT

Definition: We define a solution π as *stable* iff there doesn't exist a strategy π_S for any subset of agents $S \subseteq N$, $S \neq \emptyset$ such that $u_S(\pi_S, \pi^*_{N \setminus S}) \geq u_S(\pi)$ and $\exists i \in S : u_i(\pi_S, \pi^*_{N \setminus S}) > u_i(\pi)$. $\pi^*_{N \setminus S}$ is the stable solution to the smaller planning game over the set of agents $N \setminus S$ formed by fixing S's strategy to π_S. If $(\pi_S, \pi^*_{N \setminus S})$ is not a valid plan then we assume $u_S(\pi_S, \pi^*_{N \setminus S}) = 0$.

Note that this reduced planning problem is strictly smaller than the previous problem so eventually the non-deviating set will be reduced to $\emptyset$ at which point the definition becomes trivial.

[1] We consider asynchronous actions here and leave the concurrent action case for another paper.

4. USING SINGLE–AGENT PLANNERS

For a CoPG Π, its related centralised planning problem is $\Pi' = \langle P, \cup_{i \in N} A_i, I, \cup_{i \in N} G_i \rangle$ where each action in $\cup_{i \in N} A_i$ is given cost $c(a)$. Solving the centralised version of a CoPG leads to a plan that achieves each agent's goals. Solving it *optimally* produces the social welfare maximising plan, but not necessarily a stable solution.

We add the numerical state variables `(i-cost)` representing the cost of the joint plan so far for agent i for each $i \in N$ and update action effects with appropriate functions. Given a CoPG Π, let Π' be the centralised transformation of the problem. Also let $\Pi'|_i$ be the single-agent transformation of problem $\Pi|_i$, i.e. agent i's local planning problem involving only its own action and goal sets. We can apply the following algorithm to attempt to find a stable solution to Π:

input CoPG Π over agents N
for all $i \in N$
  for all $a \in A_i$
    append `increase((i-cost), c(a))` to add(a)
  π = the solution to $(\Pi'|_i)$
  append $(i\texttt{-cost}) \leq c(\pi)$ to G_i
  append $(i\texttt{-cost}) = 0$ to I
construct Π' from Π
output the solution to Π'.

5. SAFE–COPGS

The above algorithm does not guarantee outputting a stable solution. However, it is successful on certain empirically tested domains. The following definition captures the property that causes the above algorithm to output stable solutions.

Definition: A CoPG is *safe* iff for all possible plans π and $\forall S, S' \subseteq N$ with $S \cap S' = \emptyset$, $(\pi^*_S, \pi^*_{S'})$ is a valid plan.

Theorem: For a safe-CoPG Π, the output of the algorithm above with input Π is *stable*. (Proof omitted due to space constraints).

6. A MULTIAGENT ALGORITHM

In heuristic planners like metric-FF [4], heuristic values are calculated for each possible state by solving a relaxed version of the planning problem using planning graphs. A planning graph is a directed layered graph that contains nodes for actions and states. For each time step there is a fact layer and an action layer. At layer i the fact layer consists of all facts that can possibly be reached in i time steps and the action layer consists of all actions that are possibly applicable given those facts.

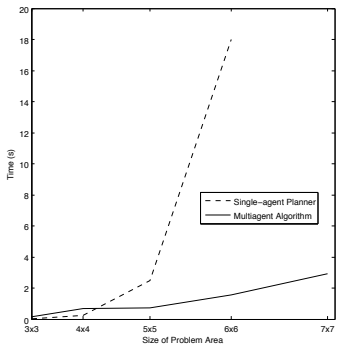
In our proposed algorithm, each agent builds an internal planning graph that consists only of facts and actions that can be performed by that agent alone. Each agent also builds a public planning graph, which includes facts added by all agents (in practice, only other agent's public facts need to be added). The plan that will be extracted depends on whether agents achieve their goals using their internal or public planning graph first. Once all goals are reached, a check is made to ensure that each agent that has only reached their goal in their public planning graph does not rely upon actions provided by agents who reached their goals on their internal planning graph first. If this fails, then planning graph generation continues until the check passes.

It may happen (even in a safe–CoPG) that one agent cannot reach its goal while constructing its internal relaxed planning graph. In this case, the only possible solution is for the agents to cooperate. If an agent's goal is unreachable in it's internal planning graph, then the agent that first achieves its goal in the public graph is forced to cooperate even if it achieved its goal internally first.

Most of the techniques utilised in metric-FF for efficient implementation [4] carry over to our algorithm. The main difference lies in the extraction of a joint plan from the joint planning graphs of multiple agents. In this case, when an agent performs an action that has a precondition provided by another agent, it adds the preconditions as a goal to the other agent's goal set.

7. RESULTS

The algorithm was evaluated in a simple grid-world parcel domain. Agents can *move* to any adjacent square, *pickup* and *drop/deliver* parcels. All actions have cost 1. The parcel domain is a safe-CoPG, since it is never beneficial to pickup another agent's parcel (the only way to potentially hinder their plan) unless planning to cooperate. The algorithm was compared, in terms of CPU time, to a single-agent planner run over the same problems on the same machine. For each different grid size, the planner was run on 100 problems and the average time taken to solve them was recorded.



On the largest problems tested, the average time taken by metric-FF (not shown on the graph) was 311 seconds while our multiagent algorithm took 2.93 seconds on average. In all cases that required cooperation, the enforced hill-climbing search performed by metric-FF failed, which effectively render its otherwise powerful heuristics useless for this kind of problem. In all 500 cases tested, the multiagent algorithm returned a stable solution.

8. REFERENCES

[1] Ronen I. Brafman and Carmel Domshlak. From one to many: Planning for loosely coupled multi-agent systems. In *ICAPS*, pages 28–35, 2008.

[2] Ronen I Brafman, Carmel Domshlak, Yagil Engel, and Moshe Tennenholtz. Planning games. In *IJCAI*, pages 73–78, 2009.

[3] Jörg Hoffmann. FF: The fast-forward planning system. *AAAI*, 22:57–62, 2001.

[4] Jörg Hoffmann. The metric-ff planning system: translating "ignoring delete lists" to numeric state variables. *JAIR*, 20(1):291–341, 2003.

[5] Raz Nissim, Ronen I. Brafman, and Carmel Domshlak. A general, fully distributed multi-agent planning algorithm. *AAMAS*, May 2010.