All for One or One for All? Distributed Planning for Selfish or Cooperative Agents

Raz Nissim

Department of Computer Science,
Ben-Gurion University
raznissim@gmail.com

Abstract

Work in both Multiagent systems and in domain independent AI has made substantial progress in recent years. For Multiagent planning, on the other hand, only very recently has a generic model that extends the single-agent STRIPS model with game-theoretic constructs been introduced, and a practical algorithm is yet to be introduced. Constructing a general, fully distributed MA planning algorithm for cooperative planning (paper to be published in AAMAS 2010) was our first step towards creating a fully distributed planner for MA systems, in which agents are self-interested, but must cooperate in order to achieve their goals. Such systems are realistic representations of real-world MA systems, whose exploration is of great interest. Distributed planning algorithms could also pave the road to effective parallelization techniques for scaling up MA planners, an area of growing interest in the planning community.

Introduction

A **MA planning problem** can be defined as follows: *Given a description of an initial state, a set of global goals, and a set of agents such that each agent has its own capabilities (operators) and private goals, find a plan for each agent that achieves its private goals, so all plans are coordinated and the global goals are achieved.* Intuitively, one would distinguish between two parts of solving such a problem: planning and coordination.

Why, one might ask, should planning and MA planning be studied separately? Is MA planning covered in the discussion of planning? In some cases, the answer is yes. A simple algorithm for the MA planning problem would have all agents upload all their information to a single agent, which would run a state-of-the-art centralized planner and broadcast its solution to all agents. In real-life problems, agents might have privacy restrictions and would object to sharing all their information. Real-life agents, who may be taxi drivers or airline pilots, may be self-interested, with their financial benefit in mind. The centralized approach would work as long as agents aren't dependent on one another (each can achieve its private goals without outside help), but if dependencies between the individual tasks exist, planning independently can result in conflicts. In this case, the answer to our question whether MA planning is covered in the discussion of planning is *no*. These inter-dependent agents must ***plan and coordinate*** their individual plans, to form a non-conflicting global plan which achieves all goals.

Background

The Model

We consider a classical MA planning setting where agents act with complete information and actions are deterministic. The problems considered are ones that can be expressed in MA-STRIPS (Brafman, Domshlak 2008), a minimalistic MA-extension of the STRIPS language. We use MA-STRIPS since it can easily be extended with aspects such as time, resources, preferences, etc.

Definition 1: *A* **MA-STRIPS Problem** *for a system of agents* $\Phi = \{\varphi_i\}_{i=1}^{k}$ *is given by a quadruple* $\Pi = \langle P, \{A_i\}_{i=1}^{k}, I, G \rangle$*, where:*

- P is a *finite set* of propositions, $I \subseteq P$ encodes the initial state, and $G \subseteq P$ encodes the goal conditions,
- For $1 \leq i \leq k$, A_i is the set of actions that the agent φ_i is capable of performing. Each action $a \in A_i$ has the standard STRIPS syntax and semantics, that is, $a = \langle Pre(a), Add(a), Del(a) \rangle$ is given by its *preconditions, add effects* and *delete effects*.

Such an MA-STRIPS problem Π induces dependencies on the agents Φ. In what follows, we use $vars(a)$ to denote $pre(a) \cup add(a) \cup del(a)$ and $effects(a)$ to denote $add(a) \cup del(a)$. Let $P_i = \bigcup_{a \in A_i} vars(a)$ be the set of all atoms affected by and/or affecting the actions of agent φ_i. By *internal* and *public* propositions of φ_i we refer to $P_i^{int} = P_i \setminus \bigcup_{\varphi_j \in \Phi \setminus \{\varphi_i\}} P_j$ and $P_i^{pub} = P_i \setminus P_i^{int}$ respectively. That is, if $p \in P_i^{int}$, no other agents can require or affect p. Using this definition of internal propositions, we can derive the partition $A_i = A_i^{int} \cup A_i^{pub}$ of agent φ_i's actions into *internal* and *public actions* respectively. That is, A_i^{int} is the set of all actions whose description contains only internal atoms of φ_i, while all other actions of φ_i are public.

To illustrate this important partition, we use the Logistics domain. Here, since all vehicle locations are internal propositions, all the *move* actions are certainly internal to the respective vehicle agents. On the other hand, *load*/*unload* actions are public just if they affect the position of a package in some of its public locations, i.e., locations that can be reached by at least two agents. Given φ_i's action a, the projection of a onto φ_i's private propositions is denoted as $a|_{int} = \langle pre(a) \cap P_i^{int}, add(a) \cap P_i^{int}, del(a) \cap P_i^{int} \rangle$. If $a \in A_i^{int}$ then $a = a|_{int}$. Otherwise, $a|_{int}$ might have fewer propositions. For example, the public action *load(p, tr, loc)*, where *loc* is a public location, has the following preconditions: *at(p, loc)* and *at(tr, loc)*. It's internal projection, however, would require only *at(tr, loc)*, since *at(p, loc)* is a public proposition. The external projection of a onto φ_i's public propositions, $a|_{ext}$, is defined analogously.

The *agent interaction graph* (AIG) IG_Π plays an important role in determining how loosely-coupled a problem is. The nodes of IG_Π correspond to the system's agents Φ. A directed edge from node φ_i to node φ_j exists in IG_Π if there exist actions $a_i \in A_i$ and $a_j \in A_j$ such that $effects(a_i) \cap pre(a_j) \neq \emptyset$. In other words, an edge from φ_i to φ_j indicates that φ_i either supplies or destroys a condition required by an action of φ_j. For example, we would have a directed edge between an airplane and a truck if the airplane unloads package p in a location reachable by the truck, since preconditions for loading p are supplied by the airplane. Edges in both directions between two agents are possible.

Planning as CSP+Planning

In (Brafman, Domshlak 2008), the authors (BD) introduced the *Planning as CSP+Planning* methodology, which solves the MA planning problem by separating planning and coordination, or the private and public aspects of the problem.

The **coordination stage**, or public aspect of the problem, refers to the agents' public actions and their execution times. Given a sequence of its public actions and their execution times, every agent must make sure that 1) every action's public preconditions are true before it is executed and 2)the goal is true in the end. The consistency of these relaxed (private preconditions are ignored) public plans is formulated as a CSP, in which each agent has a single variable, and the possible values of this variable are sequences of public actions of bounded size. The constraints in this CSP express the two consistency requirements between these action sequences, but restricted to public propositions only. Specifically, for action a with precondition p, to executed at time t, the following constraints must be satisfied: 1)there exists an action a' to be executed at time $t' < t$, which achieves p, and 2)no action a'' which destroys p is executed at time t'' such that $t' \leq t'' \leq t$. Goals also induce constraints - every goal must be satisfied after all actions are executed.

The **planning** stage, or the (agent's) private aspect of the problem, must now ensure that the agent can actually execute these public actions in a sequence. Formally, it can be viewed as a unary constraint on each of the variables in the CSP defined above. That is, it restricts the public action sequences of each agent to be locally consistent, meaning that these sequences can be extended with internal actions to ensure that the internal preconditions of each action are satisfied as well. Note that this is indeed a unary constraint.

The whole process is wrapped in an iterative deepening type search which gradually increases the upper bound, δ, on the number of public actions of each agent, along with the global plan for the whole system. These public actions serve as the coordination points between the agents. In tightly-coupled systems, one would expect the number of coordination points to be large, and this entire process is likely to be inefficient. In truly distributed and loosely-coupled systems, one would expect δ to be small, potentially increasing the efficiency of the process.

So far, relying on the formulation of BD, we described a centralized algorithm for solving the MA planning problem. As BD note, by using a *DisCSP* solver to handle the coordination part, we readily obtain a distributed planning algorithm. The high-level skeleton for this algorithm is depicted below.

```
procedure MA-Planning(Π)
  δ := 1
  loop
    Construct $DisCSP_{\Pi;\delta}$
    if ( solve-csp($DisCSP_{\Pi;\delta}$) ) then
      Reconstruct a plan $p$ from solution
      return $p$
    else
      δ := δ + 1
  endloop
```

BD's work proves the *soundness* and *completeness* of this algorithm as well as tractability under certain conditions. Moreover, obtained plans are locally optimal, since the iterative δ-loop ensures that the maximal number of coordination points between agents is minimized, optimizing plans in this sense. The algorithm's main advantage is its scalability. If we plan in a centralized manner, adding agents to a system increases complexity exponentially. Here, BD showed that if adding agents to the system doesn't change its coupling level, complexity is increased only polynomially.

A General, Fully Distributed Multi-Agent Planning Algorithm (to be Published as a Full Paper in AAMAS 2010)

Our first aim was to see whether these strong theoretical results could be translated into an efficient practical algorithm. We quickly came across a few problems. First, the constraints of the coordination CSP were not binary – every precondition of a public action must be achieved by **some** agent. This constraint effectively glues together agents that do not necessarily affect one another. Also, most, if not all, DisCSP solvers assume constraints are binary, so using one as a black box would require a change in the encoding. Second, the DisCSPs created were unlike problems previously experimented on by the constraints community. These instances are very large – every variable having hundreds of thousands of possible values or more. The sheer size of these domains means that we cannot generate them (or their constraints) a priori. This has an interesting effect on the entire process of constraint satisfaction. Current methods of improving efficiency of CSP solvers, like variable/value ordering or pruning using clever backtrack methods and forward checking, all rely on knowing the domain's size and its constraints. In our case, since domains and constraints are not fully generated, these methods were useless in the sense they require more computation than they save. Using methods and techniques from the planning world, we created **new heuristics** and adapted existing DisCSP tools to better fit these instances. Furthermore, planning tools were used to reduce domain sizes (using a coordinated relaxed planning process at preprocessing to find action landmarks) and to guide agent ordering (preferring agents that achieve goals and are most constrained). In addition, the assignment of a variable is now a complex process, involving local planning and generation of domains and constraints. This meant that a **smart agent**, capable of decision making is needed for the search to work. The agent's internal planner had to be given the ability to remember plans it had found, to avoid reassigning nogood plans when backtracking.

In this work, we presented a fully distributed MA planning algorithm, called *Planning-First*, which uses an adaptation of the *Asynchronous Forward Checking* DisCSP solver (Zivan, Meisels, 2007). Experiments were conducted on benchmark planning domains, with varying coupling levels and increasing number of agents, comparing *Planning-First* to centralized planners. While our algorithm did not perform well in Logistics, a tightly-coupled domain, in Satellites and Rovers which are more loosely-coupled, *Planning-First* showed scalability beyond centralized planners (complete results appear in the paper).

Research Plan

Extending the Results of Planning-First

From results we obtained, we found that our algorithm performs well, showing scalability beyond state-of-the-art centralized planners, in solving problem instances with limited agent interaction. From our results, it is also evident that Planning-First doesn't scale up when agents are tightly-coupled. In essence, when there are many coordination points (δ values are high), the domains of the DisCSP's variables are extremely large, rendering these DisCSPs practically unsolvable. Finding methods to deal with such tough problem instances is of great interest.

Our work showed that classical DisCSP heuristics (variants of variable and value ordering that rely on mechanism like Fail-First) are overwhelmingly dominated by heuristics that are more suited for these very large, structured DisCSPs. These heuristics use knowledge obtained from the agent's local planning, to direct the search towards achieving the agent's goals, as well as to add a fail-first mechanism to the agent selection. This makes the search similar to the planning world's regression from the goal. In our work, we would like to investigate further the use of mechanisms from the world of heuristic planning, in our agent selection and value ordering heuristics. Some examples of this are 1) further use of landmarks (which are more difficult to generate in a MA setting with incomplete knowledge) and their orderings, 2) giving an agent look-ahead capabilities by allowing it to perform relaxed planning using its neighboring agents' public actions and 3) creating a "negotiation" protocol where agents ask each other to perform certain actions. We believe that these mechanisms, and resulting heuristics, could help tackle even tightly coupled problems. Furthermore, planning obtained knowledge can also be used to further filter domains, decreasing the CSP's size.

Knowing to identify and characterize these hard problems could benefit the designers of MA systems, who may want to create systems that are easy to plan for. To do this, we would like to perform more experiments, varying and increasing the number of agents and their interaction, as well as the nature of their interaction. With more empirical results, we aim to find more delicate forms of characterizing the coupling of MA systems, and by that guiding their construction.

Offline Local Planning and Reachability Analysis

One of the bottlenecks of Planning-First is the local planning of the agents. Currently, when the agent searches for an assignment of public actions to execute, it plans locally and extracts the public actions of its local plan.

When backtracking occurs, it is known that the current action sequence is inconsistent and the sequence is added to the agent's forbidden plans. Now the agent must plan again to find a different public action sequence. This process can be performed multiple times, especially in problems with high δ values, and can be expensive in terms of computation and memory (since many action sequences are stored as NOGOODs).

The following approach could be promising, and potentially make Planning-First, and factored planning in general, more effective: The idea is to compile a *pattern database* for each agent, that holds all possible public action sequences of length k that are possible. Consider local planning with A_1 and A_2 as the landmarks (public actions to be executed). We can compute all pairs of states s_1, s_2 s.t. A_1 is applicable in s_1 and A_2 is applicable in s_2 and s_2 is reachable from s_1. We can do this for all pairs of actions A, A'. Now, to check whether a local plan for a set of landmarks $A_1, \dots, A_k$ exists, we solve the shortest path problem starting from I that passes through some state in which A_1 is applicable and some state in which A_2 is applicable etc. to the goal. In a sense, each time we need to solve this shortest path problem in a graph constructed by layers corresponding to pairs of actions.

This method could be useful in two ways. If k is large enough, we have basically worked out all the local (unary) constraints of the agent. If not, we can still use the database to rule out sequences containing infeasible subsequences.

The DB could also be used as a heuristic, if the graph is too large to fully generate. For this, we can use abstraction, as long as it ensures that if a path is impossible in the abstract case, it is indeed impossible in reality.

Parallelization

There is a growing interest in the use of parallelization techniques for scaling up planning algorithms (Kishimoto, Fukunaga, Botea, 2009). Recent results (Helmert, Roger 2008) show that in some cases, even almost perfect heuristics will not prevent exploring an exponential number of search nodes. Parallelization, as an orthogonal method of speeding up search, can continue the improvement of future planners. Our fully distributed algorithm provides a natural path to parallelizing the solution of planning problems that have a natural MA structure. Since our experiments were conducted only as a simulation of a distributed system using multi-threading, we expect to get better results when the algorithm is implemented as a truly parallel planner. Such an implementation would take advantage of the fact that in loosely-coupled systems, most of the work is done locally. We are interested in seeing what effect parallelization would have on DisCSP methods, as well as finding techniques suited for parallel MA systems.

Planning for Self-Interested Agents

The fully cooperative MA planning model lacks a quality that is part of most MA systems. A system comprising of agents that are *self-interested* seems to be a better representation of a real-world MA system. In many realistic settings, agents have personal goals and costs and are motivated to increase their net benefit. These agents may want to cooperate with one another since they have different capabilities or they find such cooperation beneficial. Such a model was described in (Brafman, Domshlak, Engel, Tennenholtz, 2009). This minimal extension of MA-STRIPS to self-interested agents, called *coalition planning games* (CoPG), describes a setting where each agent has a private goal, reward for achieving this goal and costs for its actions. Agents are self-interested, but must cooperate in order to achieve their goals. Here, a solution must ensure that no subset of agents will deviate from their plans, therefore adding some notion of stability, which introduces a game-theoretic flavor into MA-planning. Under the assumption that the AIG is acyclic, this work shows that planning for CoPG is tractable and presents an algorithm for stable planning for these systems.

It is of great interest to investigate methods for solving problems of CoPGs where the AIG is **cyclic**. These problems better represent realistic MA systems, where even cooperative agents have their own interests in mind. We believe that in order to construct an efficient algorithm for these problems, one could use methods and techniques similar to what we used in constructing *Planning-First*. Such an algorithm would bring us one step closer towards distributed planning for realistic MA systems, where agents are self-interested but willing to cooperate.

References

Brafman, R. I., and Domshlak, C. 2008. *From One to Many: Planning for Loosely Coupled Multi-Agent Systems*. In Proc. of the 18th ICAPS, 28-35.

Kishimoto, A., Fukunaga,A. and Botea, A. 2009. *Scalable, Parallel Best-First Search for Optimal Sequential Planning*. In Proc. of the 19th ICAPS, 201-208.

Helmert, M. and Roger, G., 2008, *How Good is Almost Perfect?*, In 23rd AAAI.

Brafman, R. I., Domshlak, C., Engel, Y. and Tennenholtz, M. 2009, *Planning Games*, IJCAI.

Jennings, N. R., 1996. *Coordination techniques for artificial intelligence*. In Foundations of Distributed Artificial Intelligence.

Zivan, R. and Meisels, A. 2007. Asynchronous Forward-checking for DisCSPs. In Constraints, 12, 131-150.