From One to Many: Planning for Loosely Coupled Multi-Agent Systems

Ronen I. Brafman and Carmel Domshlak

Abstract

Loosely coupled multi-agent systems are perceived as easier to plan for because they require less coordination between agent sub-plans. In this paper we set out to formalize this intuition. We establish an upper bound on the complexity of multi-agent planning problems that depends exponentially on two parameters quantifying the level of agents' coupling, and on these parameters only. The first parameter is problem-independent, and it measures the inherent level of coupling within the system. The second is problem-specific and it has to do with the minmax number of action-commitments *per agent* required to solve the problem. Most importantly, the direct dependence on the number of agents, on the overall size of the problem, and on the length of the agents' plans, is only polynomial. This result is obtained using a new algorithmic methodology which we call "planning as CSP+planning". We believe this to be one of the first formal results to both quantify the notion of agents' coupling and to demonstrate a tractable planning algorithm for fixed coupling levels.

Introduction

Suppose that we seek a plan for a system consisting of a cooperative set of agents, each with its own capabilities. To what extent would (centralized) planning for such a multi-agent (MA) system be harder than solving individual planning problems over the domains of each of the agents in isolation? Intuitively, the answer to this question should depend both on the actual problem in hand, as well as on the design of the MA system. Clearly, if the agents are tightly coupled, then planning for a MA system can become exponentially harder than individual, internal planning for each agent, because we must basically treat the system as a single, large entity. On the other extreme, planning for a completely decoupled system of agents will merely require solving a few independent single-agent planning problems, and would thus incur at most a linear factor over planning for the individual agents themselves. But what lies in between?

Intuitively, we would expect planning to become easier the more loosely coupled the system is, and we seek an algorithm that can take advantage of such loose coupling. However, "loose coupling" itself is a loose concept, and to concretize it, we need to identify a set of formal parameters quantifying the "coupling level" of MA systems. Then, we must show either that the worst-case time complexity of planning for such systems can be formulated in terms of these parameters, or that empirical run-time complexity of planning for such systems correlates with these parameters (or, of course, both). The former is what we set out to do in this paper.

A discussion of planning problems and their complexity must occur within some formal model. In this work we consider a minimalistic state-transition model expressed via the STRIPS classical planning language (Fikes & Nilsson 1971), slightly extended to associate actions with agents. To capture the level of interaction between agents we define and exploit the *agent interaction (di)graph* in which two agents are connected if one agent's action affects the functionality of the other agent. We show that the worst-case time complexity of planning for a MA system can be tied to the *tree-width* of this graph: the lower it is, the less dependent agents are on one another when they desire to change their state.

However, as we will see, the situation is a bit more complex. Besides a dependence on the structure of the system, there is also dependence on the properties of the concrete problem the system must solve. Some problems cannot be solved without much coordination between the agents, even if each agent interacts with just a few, or even one, other agents. The latter is a problem-specific parameter that corresponds to the number of actions executed by each agent that influence or are influenced by other agents in the system. Thus, one coupling parameter, denoted by ω, roughly corresponds to a measure of the number of agents that each agent must coordinate with (because they have the potential to influence her) and the other, denoted by δ, corresponds to the number of actions involving other agents that an agent must insert into its plan.

The above two parameters are intuitive, but are insufficient to formalize the worst-case time complexity of MA planning because specific problem/domain properties may affect the cost of *single* agent planning in each domain. Thus, our result is formulated in terms of the overhead of planning for a multi-agent system as a function of planning for each single agent in isolation. As noted above, this can move from an exponential overhead to a linear one. Our main contribution is to provide an algorithm that can gracefully move between these two extremes as a function of the coupling level of the system. We provide an algorithm for planning for a MA-system that is (worst-case) harder than

planning for each of its agents in isolation by a factor exponential *only* in the tree-width of the agent interaction graph and the maximal number of coordination points an agent must have. Most importantly, the direct dependence on (i) the overall size of the planning problem, (ii) the number of agents, and the length of the joint, and even individual, plans, is only polynomial. In other words, if the coupling parameters remain fixed while the number of agents increases, the cost of planning will increase only polynomially!

In our work, we build upon, combine, and extend the ideas underlying two recent proposals for factored single-agent planning by Brafman and Domshlak (2006) and Amir and Engelhardt (2003). Our key extension corresponds to a new algorithmic methodology for planning, which we refer to as "planning as CSP+planning". In particular, in contrast to the above works on factored planning, this methodology allows us to handle efficiently MA planning problems that require arbitrary long individual agent plans, provided the number of coordination points per agent is kept fixed. Overall, this gives us some of the first tractability results for non-hierarchical MA planning, and a formal characterization of coupling level and its effect on the hardness of MA planning. Moreover, although our discussion is in terms of centralized planning, the algorithm we provide is based on solving an inherently distributed CSP. Thus, using any of the many algorithms for distributed constraint satisfaction (Yokoo 2001), one obtains a distributed planning algorithm. And the underlying ideas go well beyond the simple STRIPS action model.

The paper is structured as follows. We start by defining the basic multi-agent planning model used. This definition naturally induces the notions of private/internal vs. public actions of an agent. It also leads to the definition of the agent interaction graph of the MA planning domain. Then, in the main section of this paper we show how to solve MA planning using an enhancement of planning as CSP, which we call planning as CSP+Planning. In this context, the problem of planning with landmarks arises naturally, and we show how to reduce this problem to a standard planning problem. After describing our planning algorithm, we analyze its complexity. Following this we re-examine the algorithm and modify it somewhat to get improved complexity.

Multi-Agent "Classical Planning" Model

We consider planning for cooperative MA systems in which agents act under complete information and via deterministic actions. Specifically, we consider problems expressible in a minimalistic MA-extension of the STRIPS language (Fikes & Nilsson 1971). In particular, the problems considered here comprise the seminal automata-based multi-entity models (Moses & Tennenholtz 1995). In what follows, we formalize this extension of STRIPS, as well as some of its useful derivatives that we then employ in the problem-solving part of the paper.

Definition 1 *An* **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* atoms *(also called propositions), $I \subseteq P$ encodes the* initial situation*, and $G \subseteq P$ encodes the* goal *conditions,*
- *For $1 \leq 1 \leq k$, A_i is the set of actions that the agent φ_i is capable of performing. Each action $a \in A = \bigcup A_i$ has the standard* STRIPS *syntax and semantics, that is, $a = \langle \mathsf{pre}(a), \mathsf{add}(a), \mathsf{del}(a)\rangle$ is given by its preconditions, add effects, and delete effects.*

Clearly, MA-STRIPS reduces to STRIPS exactly when $k = 1$. For ease of presentation, we assume that the individual action sets of the agents are disjoint, i.e., no two agents share an identical action. This assumption is easy to eliminate, as we explain later in the paper.

To illustrate the MA-STRIPS model, consider the well-known Logistics domain in which a set of packages should be moved on a (possibly complex) roadmap from their initial to their target locations using a given fleet of vehicles such as trucks, airplanes, etc. The packages can be loaded onto and unloaded off the vehicles, and each vehicle can move along a certain subset of road segments. It is quite natural to model this domain using MA-STRIPS by associating an atom with each package location on the map and in the vehicles, and with each truck location on the map. The action schema are $move$, $load$, and $unload$, with the suitable parameters (e.g., $move(truck, origin, destination)$ and $load(package, truck, at\text{-}location)$). Associating each truck with an agent, we might assign to this agent all the $move$, $load$, and $unload$ actions in which it is involved. (Note that disjointness of agents' action set is not problematic here as $load(P, T, L)$ and $load(P, T', L)$ are two different actions in A.)

We now focus on *dependencies* that such a MA-STRIPS problem Π induces on the agents Φ. In what follows, we use $\mathsf{eff}(a)$ as a shortcut for $\mathsf{add}(a) \cup \mathsf{del}(a)$. Let $P_i = \bigcup_{a\in A_i} \mathsf{pre}(a) \cup \mathsf{eff}(a)$ be the set of all atoms affected by and/or affecting the actions of the agent φ_i. By **internal atoms** and **public atoms** of agent φ_i we refer to the subsets $P_i^{\mathsf{int}} = P_i \setminus \bigcup_{\varphi_j\in\Phi\setminus\{\varphi_i\}} P_j$, and $P_i^{\mathsf{pub}} = P_i \setminus P_i^{\mathsf{int}}$, respectively. That is, if $p \in P_i^{\mathsf{int}}$, then other agents can neither achieve nor destroy nor even require p. Clearly, the internal atoms of all the agents are pair-wise disjoint, and there might be certain atoms that are internal to no agent. In our example, all possible truck locations are atoms internal to the truck agent, while package locations are public if they can be loaded/unloaded in these locations into/from more than one vehicle.

Using this notion of an agent's internal atoms, we can now define the partition $A_i = A_i^{\mathsf{int}} \cup A_i^{\mathsf{pub}}$ of agent actions into its **internal** and **public actions**, respectively, where

$$A_i^{\mathsf{int}} = \{a \mid a \in A_i, \mathsf{pre}(a) \cup \mathsf{eff}(a) \subseteq P_i^{\mathsf{int}}\}.$$

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. In our example, all the *move* actions are certainly internal to the respective vehicle agents, while *load, unload* actions are public just if they affect the position of a package in some of its public locations. Given an action a of agent φ_i, we use $a|_{\mathsf{int}}$ to denote the projection of a onto its

private conditions, that is, $a|_{\mathsf{int}} = \langle \mathsf{pre}(a) \cap P_i^{\mathsf{int}}, \mathsf{add}(a) \cap P_i^{\mathsf{int}}, \mathsf{del}(a) \cap P_i^{\mathsf{int}} \rangle$. If $a \in A_i^{\mathsf{int}}$, then $a = a|_{\mathsf{int}}$, but otherwise $a|_{\mathsf{int}}$ may have fewer conditions.

Finally, we introduce the notion of **agent interaction digraph** IG_Π that plays a key role in the algorithmic part of the story. The nodes of IG_Π correspond to the system's agents Φ. There is a directed edge from node φ_i to node φ_j in IG_Π if there exist actions $a_i \in A_i$ and $a_j \in A_j$ such that $\mathsf{eff}(a_i) \cap \mathsf{pre}(a_j) \neq \emptyset$. That is, an edge from φ_i to φ_j indicates that φ_i either supplies or destroys a condition required by φ_j. It is possible, of course, that there are edges in both directions between φ_i and φ_j.

It worth noting the connection between agent-interaction graph and the well known *causal graph* which plays an important role in the work of Brafman and Domshlak (2006) on factored planning. The nodes of the causal graph correspond to domain variables, and an edge connects node p to q if there exists an action a such that $p \in \mathsf{pre}(a)$ and $q \in \mathsf{eff}(a)$. Thus, the causal graph is a special instance of the agent-interaction graph when each agent is associated with a proposition and its set of actions contains all actions that influence the value of this proposition.

Planning as CSP+Planning

We now proceed to consider the algorithmic alternatives for solving a given MA-STRIPS problem $\Pi = \langle P, \{A_i\}_{i=1}^k, I, G \rangle$. Obviously, one can simply compile it into an equivalent "single-agent" STRIPS planning problem $\langle P, A, I, G \rangle$ and apply some state-of-the-art algorithm for this task. This compilation, however, hides away the original problem decomposition induced by the agents Φ. In particular, the worst-case time complexity of solving Π this way is independent of the structure and some other properties that may naturally be induced by the agent coalition Φ over the planning problem in hand. Specifically, the worst-case time complexity of leading approaches to STRIPS planning is either unbounded (for local search procedures), or exponential in the size of the problem description (for standard planning-as-CSP approaches), or exponential in the length of the shortest plan (for BFS-style procedures). The exceptions would be only some recently-proposed algorithms for factored planning (Amir & Engelhardt 2003; Brafman & Domshlak 2006; Kelareva *et al.* 2007) that we build upon in our work here. The MA-STRIPS solving framework we propose here combines some technical ideas underlying two such factored planning algorithms of (Brafman & Domshlak 2006) and (Amir & Engelhardt 2003), and extends them to target *loose agents' coupling*, which we believe to be a natural property of practical MA systems.

Coordination-Centric Planning

Consider some plan ρ for Π and an agent φ_i involved in it. Let the *individual sub-plan* ρ_i of φ_i be the order-preserving projection of ρ onto φ_i's set of actions A_i. Let $a_{i_1}, \ldots, a_{i_m}$ be the public actions in ρ_i (in their order of appearance); between each adjacent actions $a_{i_j}, a_{i_{j+1}}$ we have a (possibly empty) sequence of internal actions of φ_i. While, in principle, it is possible that the agent has no internal actions, one would expect to encounter many such actions in a system of substantially autonomous agents. Thus, we can view each agent's plan as a sequence of *coordination* (or *commitment*) points, i.e., points in which it executes actions that possibly influence or are influenced by other agents directly, and in between them, actions that do not affect other agents directly.

As an example, consider the Logistics domain described earlier, and recall that *move* actions are internal to the vehicle-operating agents. If the vehicles move on a complex map, requiring many map-point to map-point movements in between *load* and *unload* actions, then between every two actions requiring coordination, there would be many internal *move* actions. Another example would be the Rover domain that modelsaaa NASA's exploration rovers (Bresina *et al.* 2002). Imagine a set of rovers that explore a particular region. The public actions would be actions that carry out an experiment at a location, such as taking a measurement or a photo. These actions are public because they affect (goal) propositions that can be affected by many other rovers (e.g., some other rovers can also take these measurements or pictures). However, the individual plans of the rovers consist mostly of actions like moving from one location to another, tracking an object, extending the arm, warming up devices, placing instruments, calibrating instruments, etc. All these are internal actions that affect only the rover's internal state, and typically many of them come in between each pair of public actions.

Given this expectation from the MA planning problems, a promising idea should be to shift emphasis to the coordination points in the search, and let the agents "fill-in the details" on their own. In fact, this intuitive principle is already adopted one way or another in many *domain-specific* multi-agent (and, in particular, multi-robot) systems (Durfee 1999). Likewise, this principle lies in the heart of (both domain-specific and general-purpose) hierarchical planning systems (e.g., (Erol, Hendler, & Nao 1994; Knoblock 1994; Clement, Durfee, & Barrett 2007)). Our objective here is to operationalize this principle in a generic, domain-independent manner in systems that do not necessarily exhibit substantial hierarchy among the agents. We now explain how this works.

First, *suppose* that we know how many coordination points each agent requires in order to solve the planning problem. In that case, we can

(1) guess how these coordination points look like, that is, what public actions are executed in them and when, and

(2) for each agent, add internal actions between its coordination points to provide their respective internal preconditions, obtaining a legal joint plan for the system.

The latter task requires each agent to plan "in-between" its coordination points, adding internal actions that take it from the state following one public action to a state in which the next public action can be executed. In addition, different agents' individual sub-plans must be consistent—if an agent sub-plan calls for executing an action that requires some precondition to hold, then (i) either this or another agent must produce this precondition in time, and (ii) no agent is al-

lowed to destroy this precondition. For example, if the agent of truck T decides to load a package P in location L, then P should either be in L from the beginning, or be somehow brought to L in time, and, in any case, no other vehicle should be allowed to grab this package from L before T.

Of course, we have no way of guessing correctly how many coordination points there are and what their content is. On the other hand, we can try searching over all possible guesses, checking whether a guess can be extended into a complete plan. The time complexity is directly related to this: to find coordination points we perform iterative deepening over the number of coordination points, which requires time exponential in the number of coordination points. If we are not careful, a naive such iterative deepening is exponential in the total number of coordination points among all agents; that would be very problematic as this parameter is expected to grow at least as fast as the number of agents in the system. The good news is that, with care, we can reduce the time complexity to be exponential only in the number of coordination points required by a *single* agent. This number will be dominated by the agent that requires the most coordination points, and of course, we will seek to minimize this number. Note that this parameter is problem-specific because it depends on both the initial state and the goal of the MA system.

CSP and (Intra-Agent) Planning with Landmarks

We now describe a concrete procedure for extending a choice of coordination points into a globally consistent plan that corresponds to a certain *combination of constraint satisfaction and planning*

In general, a constraint satisfaction problem (Dechter 2003) is defined via a set of variables, $U = \{u_i\}_{i=1}^n$, with respective domains $\{D_i\}_{i=1}^n$, and set of constraints $\{c_i\}_{i=1}^m$. Each constraint c_i is associated with a subset of variables $\{u_{i_1}, \ldots, u_{i_{l(i)}}\}$, and defines a subset of tuples $C_i \subseteq D_{i_1} \times \cdots \times D_{i_{l(i)}}$ to be the set of allowable joint assignments to these variables. An assignment $\langle\{\theta_1, \ldots, \theta_k\}\rangle$ to U is a satisfying assignment if its projection to the domain of each constraint satisfies that constraint, that is, if $\langle\theta_{i_1}, \ldots, \theta_{i_{l(i)}}\rangle \in C_i$.

Now, assume that we allow each agent at most $\delta \geq 0$ coordination points. Thus, the total number of coordination points across the system is at most $k\delta$ (recall that k is the number of agents). Given this explicit constraint on solving Π, we define a constraint satisfaction problem $\mathsf{CSP}_{\Pi;\delta}$ over k variables $U = \{u_i\}_{i=1}^k$, one for each agent φ_i. Each such variable, u_i, represents the agent's choice of coordination points. That is, its domain consists of different choices the agent could make for the choice of coordination points. Each such choice consists of an action to execute and a time to execute this action. Thus, it is most convenient to view u_i as a vector of length δ, representing a sequence of δ coordination points. Each entry in this vector is either empty (because the agent may need fewer than δ coordination points), or is assigned a pair of the form (a, t), where a is public action of φ_i, and $t \in \{1, 2, \ldots, k\delta\}$ is an abstract time point at which φ_i commits to performing a.[1]

Our next step is to pose constraints on u_i such that any solution to the CSP defined by these variables and constraints can be extended into a legal plan for Π just if there exists a legal plan satisfying the explicit δ-bound on each agent coordination. To make things simpler and more uniform, we assume the existence of some dummy agent that has a pair of actions producing the initial state at abstract time 0, and consuming the goal state at abstract time $k\delta + 1$.

Our first constraint takes care of verifying the consistency of an agent's commitments with those of other agents affecting its non-private values.

(C1) Coordination Constraint.
An assignment $\langle\theta_1, \ldots, \theta_k\rangle$ to U satisfies C1 iff, for $1 \leq i \leq n$, $(a, t) \in \theta_i$ implies that, for each public precondition $p \in P_i^{\mathsf{pub}}$ of a holds

- for some u_j, and some $(a', t') \in \theta_j$, holds $p \in \mathsf{add}(a')$ and $t' < t$, (i.e., "someone supplies p before t") and
- for no u_l we have $(a'', t'') \in \theta_l$ with $p \in \mathsf{del}(a'')$ and $t' \leq t'' \leq t$ (i.e., "no one destroys p between t' and t").

For example, if u_T represents a truck T and $a = load(P, T, L)$, then, if (a, t) appears in the sequence of u_T, then either the agent of T or some other agent should make sure that package P gets to location L at $t' < t$, and no other agent picks it up from there within the corresponding abstract time interval $[t', t]$.

Our second constraint is posed over the internal part of the coordination-point actions to ensure that the agent is capable of supporting its own commitments. That is, the agent must be able to generate internal actions ensuring that the internal preconditions of the (public) actions it has committed to are achieved, and in the right order. To specify this constraint, we begin by formalizing a special type of single-agent planning problem which we call a STRIPS *problem with action landmarks*.

Definition 2 *A* **STRIPS problem with action landmarks** *is given by a tuple* $\Pi_L = \langle P, A, I, G, \sigma\rangle$ *where*

- P, A, I, *and* G *have the standard* STRIPS *semantics of atoms, actions, initial state, and goal, respectively.*
- $\sigma = \langle a_1, \ldots, a_{|\sigma|}\rangle$ *is a sequence of action instances from* A'*, where* A' *is defined (similarly to* A*) in terms of* P*.*

A sequence ρ *of actions from* $A \cup A'$ *is a plan for* Π_L *just if (i)* ρ *is a plan for the regular* STRIPS *problem* $\langle P, A \cup A', I, G\rangle$*, and (ii)* σ *is a subsequence of* ρ*.*

Informally, our objective in a STRIPS problem with action landmarks is to solve it in the standard sense while ensuring that the solution contains a certain sequence of actions. This sequence of actions may be disjoint from the regular actions in A, though it does not have to be. In our case the

[1] The "abstractness" of time points is crucial, but it is explained and motivated later. For now, the reader may consider these as regular time points on some discrete scale.

actions of σ will be projections of public actions onto their internal preconditions. Note that planning with action landmarks is meaningful even in the absence of a clear end-goal G. In fact, this is exactly our usage of planning with action landmarks in the specification of the *internal-planning constraint* below.

(C2) Internal-Planning Constraint.
An assignment $\langle \theta_1, \ldots, \theta_k \rangle$ to U satisfies C2 iff, for each $\theta_i = \langle (a_1^{\theta_i}, t_1), \ldots, (a_\delta^{\theta_i}, t_\delta) \rangle$, the STRIPS problem with action landmarks

$$\langle P_i, A_i^{\mathsf{int}}, I \cap P_i, \emptyset, \langle a_1^{\theta_i}|_{\mathsf{int}}, \ldots, a_\delta^{\theta_i}|_{\mathsf{int}} \rangle \rangle$$

is solvable.

Notice that C2 induces a set of unary constraints over U— it constrains each agent's coordination-point sequence in isolation, and it does not depend on the actions of other agents. However, unlike typical unary constraints, these are *procedural* unary constraints over each u_i in the form of a *single-agent planning problem* of a certain form. We now see clearly how CSP and planning are combined – CSP is used to ensure the inter-variable consistency, while planning is used to ensure intra-variable consistency (i.e., legal values for each u_i).

Putting things together, the high-level skeleton of our algorithm for MA planning problems is depicted below.

procedure MA-planning (Π over agents $\varphi_1, \ldots, \varphi_k$)
  $\delta := 1$
  loop
    Construct $\mathsf{CSP}_{\Pi;\delta}$ over $u_1, \ldots, u_k$.
    if (solve-csp($\mathsf{CSP}_{\Pi;\delta}$)) **then**
      Reconstruct a plan ρ from a solution for $\mathsf{CSP}_{\Pi;\delta}$.
      return ρ
    else
      $\delta := \delta + 1$
  endloop

The MA-planning algorithm performs an infinite loop. Each iteration, it increments the (upper-bound on the) length δ of the coordination sequences. Within the loop, the algorithm constructs the constraint satisfaction problem $\mathsf{CSP}_{\Pi;\delta}$ along the constraints C1 and C2, and checks its satisfiability. Flow-wise, this algorithm is similar to the iterative-depening algorithm for (single-agent) factored planning of Brafman and Domshlak (2006), with the (as shown below, crucial) difference being in the constraint satisfaction problems checked within the loop. Theorems 1 and 2 provide the correctness properties of the algorithm.

Theorem 1 (Soundness) *Given a* MA-STRIPS *problem* $\Pi = \langle P, \{A_i\}_{i=1}^k, I, G \rangle$*, and an upper bound* δ *on the number of coordination points per agent, if an assignment* $\langle \theta_1, \ldots, \theta_n \rangle$ *is a satisfying assignment to* $\mathsf{CSP}_{\Pi;\delta}$*, then it can be extended into a legal plan for* Π*.*

Theorem 2 (Completeness) *Given a solvable* MA-STRIPS *problem* Π*, there exists* $\delta \geq 0$*, such that* $\mathsf{CSP}_{\Pi;\delta}$ *is solvable.*

The proof of Theorem 1 requires taking care of numerous technical details, but conceptually it is quite straightforward. Satisfaction of the planning-based constraint C2 implies "conditional" validity of individual agents' plans, while these conditions are verified by the constraint C1. The latter corresponds to the standard partial-order causal-link (POCL) constraints of flaw prevention, while the standard ordering constraints of POP are replaced with associating actions with explicit time points (as is done, e.g., in temporal POCL algorithms such as CPT (Vidal & Geffner 2006)). Finally, goal-achievement of the action sequence induced by $\langle \theta_1, \ldots, \theta_n \rangle$ is ensured by our schematic addition of the dummy "goal-achiving" agent. The same line of reasoning underlies the (simpler) proof of Theorem 2.

Complexity

We now proceed to consider the time complexity of the MA-planning algorithm. Informally, this complexity corresponds to the number of times we need to verify that a certain choice of coordination-sequence length forms a basis for a solution *times* the complexity of the verification process. In other words, the time complexity of MA-planning is captured by the time complexity of solving the CSP+planning problems $\mathsf{CSP}_{\Pi;\delta}$. CSPs are a well-studied problem, and we have a relatively good understanding of their complexity. The most relevant result for our purpose is that CSPs can be solved in time polynomial in the problem size, and exponential in the tree-width of the induced constraint graph (Dechter 2003). The *constraint graph* is an undirected graph whose nodes correspond to the CSP variables, and there is an edge between u_i and u_j just if both participate in some constraint c. Informally, the *tree-width* of a graph is a measure of its "cliquishness," or how tightly coupled its nodes are (Seymour & Thomas 1993). For example, the tree-width of a tree is 1, regardless of its size, whereas the tree-width of a complete graph over n nodes is n.

Let δ denote the minimal coordination-sequence length under which a solution exists. Given that, there are at most δ coordination points for each of the k agents, which might all be executed at different time points, and each such coordination point corresponds to a public action of one of the agents. Thus, the domain D_i of each CSP variable u_i of $\mathsf{CSP}_{\Pi;\delta}$ captures

$$|D_i| = \sum_{d=1}^{\delta} \binom{k\delta}{d} \cdot |A_i^{\mathsf{pub}}|^d = O((k\delta|A_i^{\mathsf{pub}}|)^{\delta+1}) \qquad (1)$$

possible coordination sequences, where the first multiplicative term within the summation captures the choice of $d \leq \delta$ time points, and the second term captures the choice of public-action sequence of length d.

The complexity of enforcing the unary internal-planning constraints C2 is $O(f(\mathcal{I}) \sum_{i=1}^k |D_i|)$, where $\mathcal{I}$ is the maximal complexity of the individual planning for each agent in Φ, and $f(\cdot)$ captures the cost of switching from regular planning. If we let D denote $\max_{i=1}^k D_i$ then this can be written as $O(f(\mathcal{I})kD)$, where D as well satisfies $D = O((k\delta|A_i^{\mathsf{pub}}|)^{\delta+1})$. Note that the C2 constraints are unary constraints and they could be enforced "offline", resulting in an equivalent CSP with reduced variable domains.

In turn, if $CG_{\Pi;\delta}$ is the constraint graph of $\mathsf{CSP}_{\Pi;\delta}$, then checking the coordination constraint C1 can be done in time $O(kD^{\omega+1})$, where $D = \max_{i=1}^{k} D_i$, and ω is the tree-width of $CG_{\Pi;\delta}$ (Dechter 2003). Hence, we can conclude:

Theorem 3 *The overall complexity of solving* $\mathsf{CSP}_{\Pi;\delta}$ *is*

$$O\left(f(\mathcal{I}) \cdot k(k\delta|A^{\mathsf{pub}}|)^{\delta+1} + k(k\delta|A^{\mathsf{pub}}|)^{\delta\omega+\epsilon}\right) \quad (2)$$

The first term of the summation is the cumulative complexity of the single-agent sub-problems, and the second term is the complexity of extending single-agent plans to a joint MA-plan, with $\epsilon = \delta + \omega + 1$ being the dominated factor in the exponent.

Finally, we would like to establish a concrete connection between the tree-width ω of the constraint graph $CG_{\Pi;\delta}$ and the topology of the MA system. In Lemma 1 below we do exactly that by connecting between the structure of $CG_{\Pi;\delta}$ and that of the agent interaction graph IG_{Π}. The implication is that this parameter can already be known to us at system design time and does not depend on the particular planning problem solved.

Lemma 1 *For any* MA-STRIPS *problem* Π*, and any* $\delta > 0$*, the constraint graph* $CG_{\Pi;\delta}$ *induced by the constraints C1-C2 is independent of* δ*, and is isomorphic to the moral graph of* IG_{Π}.

A moral graph of a digraph G is obtained by removing the edge directions, and adding an edge between each pair of (original) parents of each node of G. Sketching the proof of Lemma 1, note that the edges of the constraint graph $CG_{\Pi;\delta}$ are only due to the coordination constraints C1. Thus, there is an edge between φ_i and φ_j either (A) if φ_i has public actions affecting preconditions of some public actions of φ_j (or vice versa), or (B) if φ_i and φ_j both have public actions affecting (either positively or negatively) preconditions of (possibly different) public actions of some third agent $\varphi_l \in \Phi$. Given that, the bijective node mapping $\forall i : u_i \mapsto \varphi_i$ establishes an isomorphism between $CG_{\Pi;\delta}$ and the moral graph of IG_{Π}; edges (A) and (B) of $CG_{\Pi;\delta}$ are mapped to the original edges of IG_{Π} and the edges connecting between the nodes' parents, respectively.

Discussion

Considering the worst-case time complexity of MA-STRIPS planning as a function of the time complexity $\mathcal{I}$ of STRIPS-planning for each of the system's agents, we have shown that the former can be upper-bounded by

$$f(\mathcal{I}) \cdot \exp(\delta) + \exp(\delta\omega)$$

that is, by the

- factor $f(\cdot)$ induced by requesting each agent to plan while committing to a certain sequence of actions,
- multiplicative factor exponential only in δ, the minmax number of *per-agent* commitments, and
- *additive* (!) factor exponential only in $\delta\omega$, where ω is the tree-width of moral graph of the agent interaction graph.

Here, ω and δ provide quantitative measures of the coupling "levels" of the system in general, and of the concrete problem instance, respectively. Note that, putting aside for a moment the factor $f(\cdot)$ of intra-agent planning, the complexity of MA-planning

(1) has no direct exponential dependence on the number of agents, k,

(2) has neither direct exponential dependence on the size $|\Pi|$ of the MA planning problem, nor such dependence on the length of a joint plan for it (and this in contrast to standard planning techniques), and

(3) has no direct exponential dependence on the length of individual agent plans, in contrast to the recent factored planning techniques we build upon (Amir & Engelhardt 2003; Brafman & Domshlak 2006)).

Having read this far, the reader may rightfully comment that planning for each individual agent can already be exponential in the overall size of the problem. Indeed, if some of the domains of individual agents have size comparable to that of the whole multi-agent system, that is, $|P_i| = \Theta(|P|)$, the whole discussion of multi-agent planning complexity seems like a waste of time, as some of the individual planning problems are about as hard as the problem of planning for the entire system. In that case, treating the system as a single entity is likely to be more profitable.

More natural and interesting settings correspond to systems in which each agent's domain is not too large, and the complexity of the system stems from the existence of many such interacting agents. In such systems we would expect the number of internal atoms of each agent to be relatively small – that is, constant or $O(\log |P|)$. Now, planning for a single agent, even if exponential in $\log |P|$, is still polynomial in P. In many MA systems this appears to be the case. For example, in the Rovers domain mentioned before, individual agents are often designed to fulfill certain well-defined roles, and their internal combinatorics can naturally end-up being simple. In fact, this is one of the major promises in devising heterogeneous MA systems: "One of the powerful motivations for distributed problem solving is that it is difficult to build artifacts (or train humans) to be competent in every possible task. Moreover, even if it feasible to build (or train) an omni-capable agent, it is often overkill because, at any given time, most of those capabilities will go to waste. The strategy in human systems, and adopted in many distributed problem-solving systems, is to bring together on demand combinations of specialists in different areas to combine their expertise to solve problems that are beyond their individual capabilities." (Durfee 1999). A nice example of this approach in the context of planning and scheduling has been proposed in (Wilkins & Myers 1998), where sophisticated systems for planning and scheduling are decomposed into modules, each of which is transformed into an agent, allowing experimentation with different degrees of coupling between the planning and scheduling capabilities.

Finally, let us consider closely the planning-with-landmarks factor $f(\cdot)$; at least at first view, planning with action landmarks seems to be more complicated than standard STRIPS planning. It is easy to show, however, that

from the worst-case time complexity perspective the overhead of adding landmarks is not significant.[2] This is because any problem $\Pi_L = \langle P, A, I, G, \langle a_1, \ldots, a_\delta \rangle \rangle$ with δ action landmarks can be compiled into an equivalent, regular STRIPS problem Π by

(i) adding a single auxiliary multi-valued variable with domain $\{q_1, \ldots, q_\delta\}$,

(ii) reformulating each action landmark a_i by setting $\mathsf{pre}(a_i) := \mathsf{pre}(a_i) \cup \{q_{i-1}\}$ and $\mathsf{add}(a_i) := \mathsf{add}(a_i) \cup \{q_i\}$, and

(iii) extending the goal G to $G \cup \{q_\delta\}$.

Note that, with this simple compilation, *the state space of* Π *is only* δ *times larger than the state space of* Π_L. Thus, assuming individual planning for each agent is polynomial (in the size of the entire system description) it is easy to verify that STRIPS planning with action landmarks for each such agent remains polynomial-time as well.

To extend the algorithm to non-disjoint action sets we need to distinguish between actions that can be performed by two agents independently and actions that require true coordination at execution. The first case is the simplest – we create two copies of the action with different names and are back to the case of disjoint sets. The second case covers both actions that require joint-execution and actions that are "mutually exclusive" – in both cases the agents must execute in coordination. The interaction graph must be modified to include edges between agents that "share" such actions, and the constraints must be modified to ensure that these actions co-occur (or not) within the sequence of public actions of the corresponding agents. Naturally, the interaction graph may be denser because of such actions, and their execution requires the ability to synchronize.

Another point to note is that the MA-planning algorithm has $k\delta$ abstract time points in which public actions are taken. These time points are abstract because any number of internal actions can come between any two public actions. In essence they serve only to constrain the order of the public actions of different agents, and not as real time points. In fact, the algorithm does the most to decouple the time points used by each agent. This may be counter-intuitive, as usually we view fully synchronized systems as easier to deal with. However, here additional synchronization would actually be a burden on the planning algorithms, as it would add unnecessary constraints to the system, and would actually increase the worst-case time complexity of the algorithms. Moreover, we see that the agents need not communicate their internal plans, nor do they need to synchronize during execution time. All an agent needs to know is that the preconditions for its next public action are satisfied.

Finally, the ability to perform the planning process in a distributed manner is of great interest, and is conceptually simple in our case. The key step in our algorithm is solving an appropriate CSP. This CSP has a natural distributed formulation and any of the many (distributed) algorithms for solving distributed CSPs could be used to generate a distributed version of the MA-planning algorithm (Yokoo 2001). The particular choice of the distributed CSP algorithm would affect properties such as communication complexity, and this can be an interesting question for future work.

[2] Of course, empirically, the situation may be quite different. But by this point it should be apparent to the reader that here we focus only on formal, worst-case analysis of these issues.

Reducing the Time Complexity

Considering the worst-case time complexity of the MA-planning algorithm as captured by Eq. 2, and recalling our interest in the time complexity of MA planning mainly *as a function* of time complexity of local planning for agents, a complexity bottleneck appears to be the exponent in the tree-width of the constraint graph $CG_{\Pi;\delta}$. In what follows, we show that this bottleneck can be partly eliminated, and sometimes to a very large degree.

Considering the statement of Lemma 1, note that the tree-width of $CG_{\Pi;\delta}$ can be $\Theta(k)$ even if the tree-width of the undirected graph induced by the agent interaction graph is $O(1)$. The reason is that the coordination constraint for the agent φ_i glues together the CSP variables corresponding to all possible providers and all possible destroyers of the preconditions of public actions A_i^{pub} (cf. the use of the *moral* graph in Lemma 1). Closely considering the language used to "communicate" commitments within the coordination process imposed by solving $CG_{\Pi;\delta}$, it turns out that sometimes we can do substantially better.

Each sequence of coordination points $\theta_i = \langle (a_1^{\theta_i}, t_1), \ldots, (a_\delta^{\theta_i}, t_\delta) \rangle$ posed by agent φ_i corresponds to a set of δ announcements of the form "at time t I will perform action a^{θ_i}". Now, let $\pi_i = \max_{a \in A_i^{\mathsf{pub}}} |\mathsf{pre}(a) \cap P_i^{\mathsf{pub}}|$ be the tight upper bound on the number of public preconditions of an action of φ_i. Note that this quantity is expected to be very low; e.g., in most (if not all) standard planning benchmarks we have $\pi_i = O(1)$ (Helmert 2003). Given that, let us extend the verbosity of each coordination point from (a, t) to $(a, t, \{(j_1, t_1), \ldots, (j_{\pi_i}, t_{\pi_i})\})$ having the semantics "at time t I will perform action a, and I require agents φ_{j_l} to provide me with the (j_l-th) non-private precondition of a at time t_{j_l}, respectively." This modification of the language does not affect the internal-planning constraints, but does effect the coordination constraints that are now reformulated as follows.

(C3) Extended Coordination Constraint. An assignment $\langle \theta_1, \ldots, \theta_k \rangle$ to U satisfies C3 iff, for $1 \leq i \leq n$, $(a, t, \{(j_1, t_1), \ldots, (j_{\pi_i}, t_{\pi_i})\}) \in \theta_i$ implies that, for $1 \leq l \leq \pi_i$, if $p_l \in P_i^{\mathsf{pub}}$ is the j_l-th public precondition of a, then

- for some u_{j_l}, and some action $a' \in A_{j_l}^{\mathsf{pub}}$, holds $p_l \in \mathsf{add}(a')$ and $(a', t_l, \{\cdot\}) \in \theta_{j_l}$, and
- for no u_j we have $(a'', t'', \{\cdot\}) \in \theta_j$ if $p_l \in \mathsf{del}(a'')$ and $t_l \leq t'' \leq t$.

Intuitively, what we required were commitments that not merely demand that someone will supply some condition, but rather, explicitly name the supplier and the supply time. This may appear a bad idea: we increased the domain

of the CSP variable because there are now many more syntactically-different coordination sequences of length δ. However, this constraint also "unglues" the providers and the destroyers of each agent φ_i. The providers now need not ensure *together* that some condition is supplied, but each provider worries only about the conditions it is explicitly requested to supply. According to Lemma 2, as long as π_i is small, this formulation can buy us a lot.

Lemma 2 *For any* MA-STRIPS *problem* Π*, and any* $\delta > 0$*, the constraint graph* $CG_{\Pi;\delta}$ *induced by the constraints C2-C3 is independent of* δ*, and is isomorphic to the undirected graph underlying* IG_{Π}.

The proof of Lemma 2 is similar to that of Lemma 1, except that now there is an edge between φ_i and φ_j in the constraint graph $CG_{\Pi;\delta}$ only if φ_i has public actions affecting preconditions of an public action of φ_j (or vice versa).

Let us now consider more closely the complexity of MA-planning with the reformulated constraint satisfaction problems $CG_{\Pi;\delta}$. The domain D_i of each CSP variable u_i now captures

$$\begin{aligned} |D_i| =& \sum_{d=1}^{\delta} \binom{k\delta}{d} \cdot |A_i^{\mathsf{pub}}|^d \cdot (k^2 d)^{\pi_i} \\ =& O((k\delta |A_i^{\mathsf{pub}}|)^{\delta+1}) \cdot \delta (k^2\delta)^{\pi_i} \end{aligned} \tag{3}$$

possible coordination sequences, where the first two multiplicative terms within the summation are as in Eq. 1, and the third term captures the choice of who (k) supports when ($k\delta$) each of the π_i public preconditions of the action. In turn, the complexity of forcing the unary internal-planning constraints C2 remains exactly as before, while the complexity of checking the coordination constraints C3 can now be done in time $O(kD^{\varpi+1})$, where ϖ is the tree-width of the (undirected) agent interaction graph IG_{Π}. The overall complexity of solving $\mathsf{CSP}_{\Pi;\delta}$ is thus order of

$$f(\mathcal{I}) \cdot k(k\delta|A^{\mathsf{pub}}|)^{\delta+1} + k(k\delta|A^{\mathsf{pub}}|)^{\delta\varpi+\epsilon'} \cdot (k^2\delta)^{\pi_i\varpi+\epsilon''}, \tag{4}$$

Note that, as we already mentioned, the tree-width ϖ can be substantially lower than the (induced by C1) tree-width ω, possibly up to a reduction from $\Theta(k)$ to 1. Hence, the reduction of worst-case time complexity (indirectly) resulting from extending the agents' language of commitments from messages used in C1 to more complex messages used in C3 can be exponential in the size of the multi-agent system.

Summary

We identified two parameters that quantify the coupling level of a multi-agent planning problem. One is system dependent—the tree-width of the agent interaction graph, and the other is problem dependent—the minmax number of coordination points per agent. When these parameters are fixed, the complexity of planning scales only polynomially with the size of the system.

Our results provide novel insights into the area of problem decomposition, and they may also help guide the design of such systems. That is, if we are to allocate actions to agents, we should strive to minimize the tree-width of the resulting agent interaction graph. They also show how a special type of single-agent planning problem is used to solve multi-agent planning problems.

There are a number of natural issues for future work. Of great interest is the design of more practical algorithms guided by the theoretical insights of this paper. If based on CSPs, these would require more efficient encodings of the problem. Execution monitoring for such systems is also an interesting topic, as the use of abstract time points gives us flexibility to handle delays as well as work with asynchronous systems.

References

Amir, E., and Engelhardt, B. 2003. Factored planning. In *IJCAI*, 929–935.

Brafman, R. I., and Domshlak, C. 2006. Factored planning: How, when, and when not. In *AAAI*, 809–814.

Bresina, J.; Dearden, R.; Meuleau, N.; Ramakrishnan, S.; Smith, D.; and Washington, R. 2002. Planning under continuous time and resource uncertainty: A challenge for AI. In *UAI*, 77–84.

Clement, B. J.; Durfee, E. H.; and Barrett, A. C. 2007. Abstract reasoning for planning and coordination. *JAIR* 28:453–515.

Dechter, R. 2003. *Constraint Processing*. Morgan Kaufmann.

Durfee, E. H. 1999. Distributed problem solving and planning. In *Multiagent systems: a modern approach to distributed artificial intelligence*. 121–164.

Erol, K.; Hendler, J.; and Nao, D. S. 1994. HTN planning: Complexity and expressivity. In *AAAI*, 1123–1128.

Fikes, R. E., and Nilsson, N. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *AIJ* 2:189–208.

Helmert, M. 2003. Complexity results for standard benchmark domains in planning. *AIJ* 146(2):219–262.

Kelareva, E.; Buffet, O.; Huang, J.; and Thiébaux, S. 2007. Factored planning using decomposition trees. In *IJCAI*, 1942–1947.

Knoblock, C. 1994. Automatically generating abstractions for planning. *AIJ* 68(2):243–302.

Moses, Y., and Tennenholtz, M. 1995. Multi-entity models. *Machine Intelligence* 14:63–88.

Seymour, P. D., and Thomas, R. 1993. Graph searching and min-max theorem for tree-width. *Journal of Combinatorial Theory* 58:22–33.

Vidal, V., and Geffner, H. 2006. Branching and pruning: An optimal temporal POCL planner based on constraint programming. *AIJ* 170(3):298–335.

Wilkins, D. E., and Myers, K. 1998. A multiagent planning architecture. In *Int. Con. on AI Planning Systems*, 154–162.

Yokoo, M. 2001. *Distributed Constraint Satisfaction: Foundations of Cooperation in Multi-agent Systems*. Springer.