Complexity of Single-Agent and Equilibrium-Based Multiagent Planning and Plan Checking: Approach When We Have a List of Valid States

Chitta Baral and Vladik Kreinovich

Abstract

In a recent paper M. Bowling, R. Jensen, and M. Veloso proposed a new formalization of the problem of multiagent planning – a formalization that is based on the (intuitively natural) notion of an equilibrium. In this paper we analyze the computational complexity of the corresponding equilibrium-based multiagent planning and plan checking. Within the traditional approach in which states are described by fluents, the computational complexity of planning under incompleteness is **PSPACE**-hard already for a single agent; therefore, to make a meaningful comparison between the complexity of single-agent and multi-agent planning, we analyze complexity in a different approach, in which a list of valid states is assumed to be explicitly given.

1 Introduction

1.1 Multiagent Planning Is Important

Traditionally, planning was about planning for a single agent. For a single agent, it may be computationally difficult to find an optimal plan, it may be even difficult to check that the plan is good, but at least we know what exactly we want from the plan: that, as the result of this plan, we satisfy the desired goal.

The situation in which there is only agent is, of course, an oversimplification. There are usually multiple agents, and the action of each agent may influence the results of the other agents. It is therefore desirable to take this influence into consideration when formulating and analyzing planning problems, In other words, it is necessary to consider multiagent planning.

1.2 Multiagent Planning Is Difficult to Describe

For multiagent planning, it is not easy even to define what it means for a multiagent plan to be successful. Preliminary definitions have been discussed, e.g., in [Jensen and Veloso, 2000; Wilkins and Myers, 1998]. A very convincing definition of multiagent planning – motivated by general notion of an *equilibrium* from game theory – was introduced in [Bowling *et al.*, 2002].

The main idea behind the notion of an equilibrium is that there is no incentive for any agents to abandon the mutually agreed strategies. In other words, if one agent i replaces his original strategy A_i with some other strategy $A_i' \neq A_i$, this replacement will not improve this agent's success – and therefore, it makes sense for all the agents to stick to their pre-agreed strategies.

In game theory (see, e.g., [Owen, 1995]), success is characterized by a real number. In AI, a success is usually described in "yes"-"no" terms: whether we have achieved the goal or not. Thus, for deterministic actions, we have *two* possible values of success: success achieved and success not achieved. For a more realistic description, in which we take into consideration nondeterministic consequences of different actions, we have *three* natural levels of success:

- the lowest is when for all possible results of the actions, the agent does not achieve its goal;
- the intermediate is when for some possible result of the actions, the agent achieves its goal, but for some other possible results, the agent does not achieve its goal;
- the highest is when for all possible results of the actions, the agent achieves its goal.

(It is worth mentioning that, for the purpose of clarity, we are somewhat simplifying the descriptions from [Bowling *et al.*, 2002].)

In these terms, the equilibrium means the following:

- if, for the original set of strategies, the agent i never succeeds, then, no matter how this agent changes his strategy, this agent will never succeed;
- if, for the original set of strategies, it is possible that the agent i does not succeed, then, no matter how this agent changes his strategy, there will always be a possibility that this agent does not succeed.

1.3 It Is Important to Analyze Computational Complexity of Multiagent Planning

One of the main obstacles that prevent widespread practical applications of planning is that planning in general is known to be a computationally difficult problem; see, e.g., [Baral *et al.*, 2000; Erol *et al.*, 1995; Liberatore, 1997; Littman, 1997] and references therein. Main practical successes of planning come from considering tractable (easy-to-solve) particular cases.

Even planning for a single agent is a difficult computational problem; adding extra agents can only make this problem more complex. It is therefore desirable to analyze how exactly more difficult is the corresponding multiagent problem.

1.4 For This Analysis, We Need to Further Specify the Definitions

Before we proceed with this analysis, we need to make one important comment. The original definition of the multiagent equilibrium was given from the viewpoint of clarity and mathematical accuracy; this definition is very precise and, by itself, does not require any clarifications or modifications. However, if we want to analyze its computational complexity, we must provide further clarifications.

Let us give a simple example of why it is important. A part of the definition of the multiagent planning domain is a set of valid states $\mathcal{S}$ which is defined as a subset $\mathcal{S} \subseteq 2^{\mathcal{P}}$, where $\mathcal{P}$ is a finite set of propositions (fluents) that describe different states. There are two different natural computational interpretations of this definition:

- we can assume that we are given a *list* of valid states;
- alternatively, we can assume that we are given an *algorithm* that, given a state, checks whether this state is valid or not.

The second interpretation is a particular case of the first one: if we have a list, then, of course, we can design a validity-checking algorithm by simply comparing a new state with all the states form the given list. However, from the viewpoint of computational complexity, these two interpretations are not equivalent. For example, suppose that every state is valid, and we are interested in checking whether there exists a state for which a given propositional combination of fluents is true. Then:

- if we are given a *list* of states, then this checking can be done by simply testing all the states from the given list; the time necessary for this check is linear in the length of the input, so in this formulation, the desired checking is feasible (i.e., polynomial time);
- if, instead, we are given a validity-checking algorithm (a trivial one that returns "true" for every state), then the desired checking problem becomes the problem of propositional satisfiability – the problem known to be **NP**-hard.

(In this particular case, the examples given in [Bowling *et al.*, 2002] seem to indicate that the authors of this paper have the first interpretation in mind.)

1.5 For a Meaningful Comparison of Computational Complexity of Single-Agent and Multi-Agent Planning, We Need a Different Approach to Describing Complexity

In the traditional approach to analyzing complexity of planning, states are described by fluents. In this approach, the computational complexity of planning under incompleteness is **PSPACE**-hard already for a single agent; see, e.g., [Baral *et al.*, 2000].

Therefore, to make a meaningful comparison between the complexity of single-agent and multi-agent planning, we must analyze complexity in a different approach, when a list of valid states is explicitly given.

In this paper, we start with formulating this different approach; then, we re-visit the problems of single-agent planning in this alternative approach, and finally, we analyze the complexity of the multiagent planning in this approach.

2 Alternative Approach to Describing States When Analyzing Computational Complexity of Planning and Plan Checking

2.1 How to Describe States: Traditional Approach

We have already mentioned that even for a single agent, the planning problem is computationally difficult; in several reasonable formalizations, this problem is **NP**-hard; see, e.g., [Baral *et al.*, 2000] and references therein. These formulations are based, in particular, on the following idea of how we can describe different states.

To find out what a state is, we can perform certain measurements and/or expert estimates. As a result of each measurement, we get the values of the measured quantities; these values characterize the state. In engineering and control, we usually consider real-valued quantities; as a result, each measurement result is a real number, and a state is therefore described as a sequence of real numbers. In AI – specially in logic-based AI – it is natural to consider *fluents*, i.e., "true"-"false" characteristics describing a given state. From this viewpoint, a state can be described by a sequence of fluents. In other words, we are given a list of n fluents $f_1, \ldots, f_n$, and a state is characterized by the corresponding n truth values.

2.2 The Problem with the Traditional Approach

The problem with this description is that to get a better and better description of a state, we must add more fluents. However, for n fluents, we have 2^n possible states,

so for reasonable n, we have a very large number of possible states. In the above-mentioned NP-hardness results, we implicitly assume that all 2^n states are possible. As a result, even the problem of finding a state with given properties becomes difficult, and planning becomes **NP**-hard – or even **PSPACE**-hard.

The assumption that we have 2^n possible states means that we assume that all the fluents are independent. In reality, fluents are heavily dependent; as a result, only some of the 2^n combinations actually correspond to valid states: once the few main fluents are described, most states are described uniquely; some still need to be clarified, so we may need extra fluents, but we definitely do not double the overall number of states after introducing each extra fluent.

2.3 How to Describe States: Alternative Approach (Implicitly Used in Describing Multiagent Equilibria)

Summarizing: instead of describing states as interpretation of a set of fluents, it is reasonable to assume that we have a *list* $\mathcal{S}$ of possible states.

We also have a finite list $\mathcal{A}$ of actions. Since we consider a deterministic case, we assume that we know, for each state s and for each action a, a consequence $res(a, s) \in \mathcal{S}$ of applying this action.

2.4 This Approach Has Been Used in Planning and Multiagent Planning – Implicitly and Explicitly

Traditionally, in the analysis of computational complexity, states are described by fluents because this is what planning algorithms like STRIPS take as inputs. However, in examples that explain the commonsense meaning of planning and plan checking (both single-agent or multiagent), in many case, an explicit list of valid states is given. In other words, the alternative approach to state description is in a very good accordance with common sense and is, in this sense, implicitly used in planning.

It is also explicitly used in some papers, even in some papers that analyze the computational complexity of multiagent planning problems; see, e.g., [Pynadath and Tambe, 2002].

2.5 Computational Complexity of Planning and Plan Checking within the Alternative Approach to State Description: Case of a Simple Goal

In the traditional description of the planning problem, we use a simple description of the goals: namely, we assume that we know the initial state s_0, and that the goal is to reach a state from a given subset (sublist) $\mathcal{G} \subseteq \mathcal{S}$.

A *plan* is then defined as a sequence of actions $a_1, \ldots, a_n$. We say that a plan is *successful* if after this sequence of actions, we get a state from $\mathcal{G}$, i.e., that $res(a_n, res(a_{n-1}, \ldots (res(a_1, s_0) \ldots)) \in \mathcal{G}$. The resulting general problem of *planning for a single agent in a deterministic environment* is easy to solve:

- From the very definitions, it follows that there exists a polynomial-time algorithm that, given an instance of the planning problem and a plan, checks whether this plan is successful.
- There also exists a polynomial-time algorithm that, given an instance of the planning problem, either finds a successful plan or (correctly) reports that there is no successful plan.

(In graph terms, a plan is a path from the initial state to to one of the final states; thus, this result immediately follows from Dijkstra's algorithm.)

3 Different Approach to Describing States Requires a Different Approach to Describing Goals

3.1 How Goals are Described in Real Life

In the previous section, we followed the traditional planning approach and described goals as desired final states. In real life, a goal is described not just by the final state but also by the efforts that led to this goal. For example, when we plan the motion of an industrial robot, we not only specify where this robot must be at the end, but also that this robot should not go out of the industrial zone, and that it should not spend too much energy on its actions. In short, the actual goal depends not only on the final state, but also on the actions that led to this state and on the intermediate states.

3.2 In Traditional Description of States, The Description of Goals Can Be Simplified

In traditional description of the planning, the description of the goal is usually simplified by claiming that the goal is simply one of the final states. This simplification can be done without losing generality because we can always add history (previous actions and previous states) to our description of a state.

3.3 In a New Description of States, the Traditional Goal Simplification Is No Longer Possible: A New Description of Goals

In our new formulation of the planning problem, when the states come from a given list, we can no longer do that: if we have S states, then after T steps, we get S^T possible histories; hence, for large T, we cannot describe all the histories as a list.

As a result, for our new description of states, we cannot always use the simplified description of a goal as a final state or as a set of desirable final states. Instead, we must describe a goal as a *feasible* function (i.e., polynomially evaluable) that, given a sequence of actions $a_1, \ldots, a_n$ and a sequence of states $s_1, \ldots, s_n$, returns "true" or "false" depending on whether the goal is satisfied or not.

3.4 Some Simplification Is Still Possible

Some simplification is still possible here: since we consider a deterministic case, the states are uniquely determined by actions, so we do not need to consider states at all: it is sufficient to consider sequences of actions. In other words, we have a function $g(a_1, \ldots, a_n)$ that maps sequences of n actions into "true" or "false"; we say that a plan $(a_1, \ldots, a_n)$ is successful if $g(a_1, \ldots, a_n)$ is "true".

4 A Different Approach to Goal Description Necessitates a Different Description of a Plan: Restriction to Plans of Polynomial-Time Duration

In principle, the above definition of a plan can lead to arbitrarily long plans.

¿From the practical viewpoint, the same arguments that show that only polynomial-time computations are practically possible can be used to conclude that only polynomial-length plans are feasible. In other words, we have a (growing) polynomial $P(n)$ that describes, for each input length n, the largest possible duration of a plan. The problem is: given a planning situation whose description length is n, to produce a feasible (= of duration $\leq P(n)$) successful plan.

5 Single Agent, Deterministic Actions: Results on Computational Complexity of Planning and Plan Checking

Let us describe the computational complexity of a single-agent planning and plan checking in this new formulation.

Proposition 1. (single agent, deterministic actions, realistic goal) *There exists a polynomial-time algorithm that, given a plan, checks whether this plan is successful.*

As usual in computational complexity (see, e.g., [Papadimitriou, 1994]), by complexity of planning, we means the complexity of checking whether there is a plan.

Proposition 2. (single agent, deterministic actions, realistic goal) *The planning problem – i.e., the problem of checking the existence of a successful plan – is* **NP***-complete.*

(For reader's convenience, all the proofs are given in the special Proofs section.)

6 How to Describe Non-Deterministic Actions in This Approach

In general, the goal depends on the actions and states at different moments of time. In the deterministic case, actions uniquely determine the states, so, without losing generality, we can assume that the goal g only depends on the sequence of actions.

In real-life, the results of an action are not uniquely determined by the agents' action; they are *non-deterministic* in the sense that they depend on some external actions and properties $e_1, \ldots, e_m$ that are beyond our control. We know the set of possible values of each e_i; once we know the agent's actions a_i and the values e_j, we are able to determine all the states. We can substitute this dependence of the states s_i on a_j and e_j and thus, conclude that, in general, the goal can be described solely in terms of a_j and e_j.

In other words, we have a finite set $\mathcal{A}$ of actions, and we have a finite set $\mathcal{E}$ of possible external actions. We have a feasible function $g(a_1, \ldots, a_n, e_1, \ldots, e_m)$ that, given a sequence of actions and a sequence of external actions, returns "true" or "false".

We must also have a function that describes, for each state s, for each sequence of external actions $e_1, \ldots, e_m$, and for each action a, the resulting state s'.

7 Planning for Situations with Non-Deterministic Actions: Necessity for Conditional Actions

The possibility of external actions changes the notion of a plan. In the deterministic case, once our actions are fixed, we know what will happen to a system; so to describe a plan, it is sufficient to describe the action a_t performed at each moment t.

In non-deterministic case, we may end up with different states at the same moment of time, so it is reasonable to plan different actions depending depending on where we end up. In other words, instead of the original actions, we must consider *conditional actions*, i.e., functions from the set of all states to the set of all actions.

To describe each conditional action, we must describe, for each moment t and for each state $s \in \mathcal{S}$, the corresponding action $a_{t,s}$. Thus, a *plan* is now a sequence $a_{i,s}$, where $1 \leq i \leq n$ and $s \in \mathcal{S}$. For each plan and for each sequence of values $e_1, \ldots, e_m$ of external variables, we can compute the actual states and thus, the value of the goal function.

We say that a plan is:

- *necessarily successful* if g="true" for all possible values $e_1, \ldots, e_m$;
- *possibly successful* if g="true" for some possible values $e_1, \ldots, e_m$.

8 Single Agent, Non-Deterministic Actions: Computational Complexity of Planning and Plan Checking

Proposition 3. (single agent, non-deterministic actions, realistic goal) *Checking whether a given plan is necessarily successful is* **coNP***-complete.*

Proposition 4. (single agent, non-deterministic actions, realistic goal) *Checking whether a given plan is possibly successful is* **NP***-complete.*

Proposition 5. (single agent, non-deterministic actions, realistic goal) *Checking the existence of a necessarily successful plan is* $\Sigma_2\mathbf{P}$*-complete.*

Proposition 6. (single agent, non-deterministic actions, realistic goal) *Checking the existence of a possibly successful plan is* **NP***-complete.*

9 Description of Equilibrium-Based Multiagent Planning within the Alternative Approach to State Description

For each agent k ($1 \le k \le K$), we have a set $\mathcal{A}^{(k)}$ of possible actions. We also have a finite set $\mathcal{S}$ of possible states and a finite set $\mathcal{E}$ of possible external actions. For each agent k, we have a feasible function

$$g(a_1^{(1)}, a_1^{(2)}, \ldots, a_n^{(K)}, e_1, \ldots, e_m)$$

that, given the actions of all the agents and all the external actions, returns "true" or "false" depending on whether the goal of k-th agent is satisfied or not.

We must also have a function that describes, for each state s, for each sequence of external actions $e_1, \ldots, e_m$, and for each combination of agent's action $a^{(1)}, a^{(2)} \ldots$, the resulting state s'.

A *plan* consists of *strategies* of each agent, and each strategy is (like in the previous section) a sequence of conditional actions. Similarly to the previous section, we say that a plan is

- *necessarily successful* for agent the k if g_k="true" for all possible values $e_1, \ldots, e_m$;
- *possibly successful* for the agent k if g_k="true" for some possible values $e_1, \ldots, e_m$.

We say that a plan is a *equilibrium* if the following two conditions are satisfied for each agent k:

- if this plan is not necessarily successful for this agent, then, if the agent k changes its strategy, the result will still not be necessarily successful for this agent;
- if this plan is not possibly successful for this agent, then, if the agent k changes its strategy, the result will still not be possibly successful for this agent.

10 Multiple Agents: Computational Complexity of Planning and Plan Checking

Proposition 7. *Checking whether the existing plan is an equilibrium in a* $\Pi_2\mathbf{P}$*-complete problem.*

Proposition 8. *The problem of checking the existence of an equilibrium plan belongs to the class* $\Sigma_3\mathbf{P}$.

11 Conclusion and Future Work

In this paper, we:

- explicitly formulate and develop an alternative approach to the analysis of computational complexity of plan checking, an approach in which the valid states are given as a list;
- we then use this alternative approach:
  - to re-visit the problems of computational complexity of single-agent planning, and
  - to analyze the computational complexity of multiagent planning.

By comparing Proposition 7 with Proposition 3 and 4 (and Proposition 8 with Propositions 5 and 6), we see that our results are reasonably optimistic: Namely, we prove that although the equilibrium-based multiagent planning problem is more complex than the corresponding single agent planning problem, the increase in complexity is, in some natural sense, the smallest possible – exactly one level on the polynomial hierarchy.

Informally, *multiagent planning is not that much more complex than single-agent planning.*

Open problems: it is desirable to extend these results to even more realistic models of agents, e.g., to models of communicating agents; these models (and the corresponding computational complexity results) are described, e.g., in [Pynadath and Tambe, 2002].

12 Proofs

12.1 Proof of Propositions 1–2

The proof of Proposition 1 is straightforward.

To prove Proposition 2: the existence of a plan has the form $\exists a_1 \ldots \exists a_n\, g(a_1, \ldots, a_n)$ for a feasible property g; thus, this problem is in the class **NP**.

Vice versa, every problem from the class **NP** (e.g., propositional satisfiability) can be thus represented, so this planning problem is indeed **NP**-complete.

12.2 Proof of Proposition 5

The existence of such a plan is equivalent to $\exists a_{1,s_1} \ldots \exists a_{n,s_S} \forall e_1 \ldots \forall e_m\, g$, where s_i denotes i-th state in the listing $\mathcal{S}$ of all the states, and g is feasible; thus, this existence belongs to the class $\Sigma_2\mathbf{P}$.

To prove that it is $\Sigma_2\mathbf{P}$-complete, it is thus sufficient to prove that a known $\Sigma_2\mathbf{P}$-complete problem – checking $\exists x_1 \ldots \exists x_k \forall y_1 \ldots \forall y_l\, F$ for Boolean variables x_i and y_j and a propositional formula F – can be reduced to this planning problem.

Indeed, for this reduction, we can take the following planning problem:

- this problem has two actions, "true" and "false";
- it has two similar external actions;
- the goal function g that does not depend on the states at all – only on the actions x_i and external actions y_j, as $F(x_1, \ldots, x_k, y_1, \ldots, y_n)$.

In this case, there is no reason to even consider conditional actions, and the necessarily successful plan is possible if and only if the above quantified propositional formula is true.

12.3 Proof of Proposition 6

The existence of such a plan is equivalent to $\exists a_{1,s_1} \ldots \exists a_{n,s_S} \exists e_1 \ldots \exists e_m g$, where g is feasible; thus, this existence belongs to the class **NP**. Due to Proposition 2, already a subclass of this problem is **NP**-complete; thus, the new problem is **NP**-complete as well.

12.4 Proof of Propositions 3 and 4

Propositions 3 and 4 are proved similarly.

12.5 Proof of Propositions 7 and 8

The fact that a plan a is an equilibrium can be described as saying that for every k, the following two statements hold:

$$(\exists e_1 \ldots \exists e_m \neg g_k) \rightarrow (\forall a^{(k)} \exists e_1 \ldots \exists e_m \neg g_k)$$

and

$$(\forall e_1 \ldots \forall e_m \neg g_k) \rightarrow (\forall a^{(k)} \forall e_1 \ldots \forall e_m \neg g_k).$$

The first statement can be reduced to $\forall e\, g_k \vee \forall a^{(k)} \exists e \neg g_k$ (where $\forall e$ is an abbreviation for $\forall e_1 \ldots$) – i.e., to a statement from $\Pi_2\mathbf{P}$; the second statement can also be reduced to a statement of the same (actually, simpler) type. Thus, the plan checking problem belongs to the class $\Pi_2\mathbf{P}$.

So, the existence of such an equilibrium can be describe as $\exists a\, G$, where $G \in \Pi_2\mathbf{P}$ – i.e., by a formula from the class $\Sigma_3\mathbf{P}$; Proposition 8 is proven.

To complete the proof of Proposition 7, it is thus sufficient to prove that a known $\Pi_2\mathbf{P}$-complete problem – checking $\forall x_1 \ldots \forall x_l \exists y_1 \ldots \exists y_m F$ for Boolean variables x_i, y_j, and a propositional formula F – can be reduced to this plan checking problem.

Indeed, we can take the following planning problem:

- it has two actions, "true" and "false";
- it has two similar external actions;
- the goal function g_1 that does not depend on the states at all – only on the actions x_i and external actions y_j, as

  $$g_1 = (x_0 \,\&\, y_0) \vee (\neg x_0 \,\&\, \neg F(x_1, \ldots, x_l, y_1, \ldots, y_m));$$

- the goal functions g_k, $k > 1$, always return "true".

In this case, every plan is necessarily successful for all the agents except maybe the agent 1, so to check that the plan is an equilibrium, it is sufficient to check that this plan is an equilibrium for agent 1.

For this agent, for a sequence of actions starting with a "true" action x_0="true":

- sometimes, we have success (when y_0="true"), and
- sometimes, we do not have success (when y_0="false").

In other words, the corresponding plan is possible successful but not necessarily successful for agent 1.

The only case when the plan including this strategy of agent 1 is an equilibrium is when there is no other strategy of agent 1 that will make this agent necessarily successful – i.e., when for all possible other strategies $x_1, \ldots, x_l$, there exists a possible case $y_1, \ldots, y_m$ for which the plan is not successful – i.e., for which F is true. This condition is exactly what the original quantified propositional formula describes. Thus, we have the desired reduction, and so Proposition 7 is proven.

References

[Baral *et al.*, 2000] Chitta Baral, Vladik Kreinovich, and Raúl Trejo. Computational complexity of planning and approximate planning in the presence of incompleteness. *Artificial Intelligence*, 122:241–267, 2000.

[Bowling *et al.*, 2002] Michael Bowling, Rune Jense, and Manuela Veloso. A formalization of equilibria for multiagent planning. In *proceedings of the AAAI-2002 Workshop on Multiagent Planning*, Edmonton, Canada, August 2002.

[Erol *et al.*, 1995] K. Erol, D. S. Nau, and V. S. Subrahmanian. Complexity, decidability and undecidability results for domain-independent planning. *Artificial Intelligence*, 76:75–88, 1995.

[Jensen and Veloso, 2000] R. Jensen and M. Veloso. OBDD-based universal planning for synchronized agents in non-deterministic domains. *Journal of Artificial Intelligence Research*, 13:189–226, 2000.

[Liberatore, 1997] P. Liberatore. The complexity of the language $\mathcal{A}$. *Electronic Transactions on Artificial Intelligence*, 1:13–28 (1997) http://www.ep.liu.se/ej/etai/1997/02

[Littman, 1997] M. Littman. Probabilistic propositional planning: representation and complexity. *AAAI'97*, 748–754.

[Owen, 1995] G. Owen. *Game Theory*. Academic Press, New York.

[Papadimitriou, 1994] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, Reading, MA, 1994.

[Pynadath and Tambe, 2002] David V. Pynadath and Milind Tambe. Multiagent teamwork: analyzing the optimality and complexity of key theories and models. In *Proceedings of AAMAS'02*, Bologna, Italy, July 15–19, 2002,

[Wilkins and Myers, 1998] D. E. Wilkins and K. L. Myers. A multiagent planning architecture. In *Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems*, 145–162.