ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-1111

Generating Multi-Agent Plans by Distributed Intersection of Finite State Machines

Jan Tožička and **Jan Jakubův**[1] and **Antonín Komenda**[2]

Abstract. Deterministic multi-agent planning described by MA-STRIPS formalism requires mixture of coordination and synthesis of local agents' plans. All agents' plans, as sequences of actions, can be implicitly described by an appropriate generative structure. Having all local plans of all participating agents described by such a structure and having a merged process of such structures, we can induce a global multi-agent plan by successive elimination of unfeasible combinations of local agents' plans.

We use Nondeterministic Finite State Machines (NFSs) as the generative structure for plans and a well-known process of intersection of NFSs to prune the plan spaces towards globally feasible multi-agent plan. Since the numbers of the plans can be large, we use iterative process of building of the NFSs using a state-of-the-art classical planner interleaved with the NFS intersecting process. We have evaluated the algorithm on an extensive number of problems from International Planning Competition extended for multi-agent planning.

1 Introduction

Deterministic domain-independent multi-agent planning (DMAP) based on Brafman and Domshlak's MA-STRIPS formalism [3] is a rather young area fusing classical automated planning and multi-agent paradigm with cooperative agents which can keep some private information about the planning problem. To solve a DMAP problem, agents has to deliberate on future actions transforming an environment towards a defined goals. Intelligent agents in such environment have to be able not only to selfishly push the world towards their own goals, but also have to cooperate on common goals with other agents and solve mutual conflicts if their plans interfere. DMAP deals with challenges both in the *synthesis of individual agents plans* and in the *coordination of the agents' plans* in a shared environment.

The approach proposed in this paper can be seen as a mixture of an iterative-adaptive approach [2, 6] supported by strong coordination-centric element and also as a specialized and practical form of satisfying principle from [4] applied to the MA-STRIPS model. We propose to implicitly describe all possible plans of each agent in form of a generative Nondeterministic Finite State Machine (NFS) and by a distributed *coordinated* intersection process of those machines eliminate all unfeasible combinations of plans and therefore end up with a global multi-agent plan feasible as a combination for all agents. Since the number of all (non repeating) possible agent's plans grows exponentially with the size of the agent's planning sub-problem, the NFS is built *iteratively* with help of a forward-search planner.

[1] Czech Technical University in Prague, Faculty of Electrical Engineering, Czech Republic, email: {jan.tozicka,jan.jakubuv}@agents.fel.cvut.cz
[2] Technion – Israel Institute of Technology, Faculty of Industrial Engineering and Management, Israel, email: akomenda@tx.technion.ac.il

2 MA-STRIPS Planning

We consider a number of *cooperative* and *coordinated* agents featuring distinct sets of capabilities (actions) which concurrently plan and execute their local actions in order to achieve a joint goal. The environment wherein the agents act is *classical* with *deterministic* actions. In this paper we consider the basic MA-STRIPS concepts as defined by Brafman and Domshlak [3].

In MA-STRIPS planning, every agent is assigned a set of actions he can perform in a shared environment. An MA-STRIPS *planning problem* Π is defined as a quadruple $\Pi = \langle P, \{\alpha_i\}_{i=1}^n, I, G\rangle$, where P is a set of facts, α_i is the set of actions that identifies the i-th agent, I is an initial state, and G is a set of goals. Actions are defined as in classical STRIPS as triples of preconditions, add effects and delete effects. MA-STRIPS distinguishes between *public* and *internal* facts. A fact from P is called *public* iff it is mentioned by actions of at least two different agents. A fact from P is called *α-internal*, or *internal for α* iff the fact is mentioned only by actions of agent α. A fact is *relevant for α* iff it is public or α-internal. We restrict our research to the problems where all the goals are public. See [3] for more details.

An action of α is called *public* iff the action has a public effect. Otherwise the action is called *internal*. A *projection of action a* of agent α to agent β is the restriction of a to the facts relevant for β. In MA-STRIPS also actions with public preconditions are treated public. We can relax this restriction as our approach does not require agents to synchronize on public preconditions. For every agent α, we define α's *local planning task* Π^α as a classical STRIPS problem over the facts relevant for α where the set of actions contains all the actions of α together with the projections of all public actions of all the other agents. A *public projection* of a local solution π is obtained by restricting π to public actions. A solution of an MA-STRIPS problem Π can be obtained from local solutions of all the agents provided the local solutions have the same public projection. Thus we transform the task of solving Π to the task of finding local solutions with an equal public projection. See [6] for details.

3 Planning State Machines (PSM)

In our work we use nondeterministic finite state machines (NFS) [5] to represent a set of solutions of a STRIPS problem. We call an NFS that represents a set of solutions a *planning state machine* (PSM). For every PSM representing a set of solutions we can construct its public projection PSM that represents public projections of the solutions. Once we have a PSM for every agent's local planning task Π^α, we can compute public projections of these PSMs and intersect them using well-known intersection algorithm for NFS. Any public solution in a non-empty intersection constitutes a solution of the original MA-STRIPS problem Π.

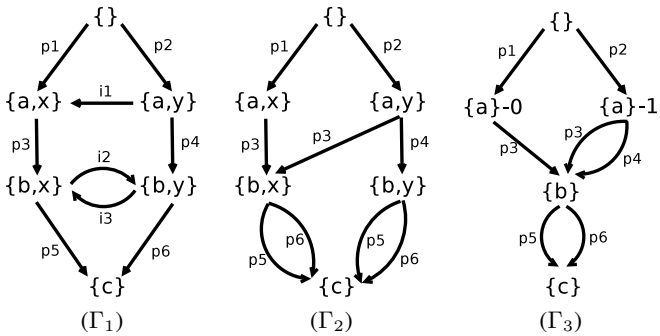

Figure 1. Example of computing PSM public projection. We suppose a context where pn are public and in internal actions, and where a, b, c are public and x, y internal facts.

A *planning state machine (PSM)* of a STRIPS problem $\Pi = \langle P, A, I, G\rangle$ is a NFS $\Gamma = \langle A, S, I, \delta, F\rangle$ where the alphabet A is the set of actions of Π, states from S are subsets of P, the state transition function δ simply resembles the classical STRIPS action application γ, and $F \subseteq S$ contains all the states that satisfies goal G. It is possible to construct a *complete PSM* that contains all possible states and transitions. A complete PSM of Π represents exactly all the solutions of Π. As the computation of a complete PSM can be inefficient, we also consider a *partial* PSM which represents only a subset of all solutions.

Figure 1 gives an example of a PSM (Γ_1) to demonstrate a PSM public projection algorithm. PSM Γ_2 is obtained from the input Γ_1 by renaming internal actions to ϵ-transitions and eliminating them by a well-known algorithm [5]. The public projection Γ_3 of Γ_1 is obtained from Γ_2 by removing internal facts and by unification of states with equal public projection. When two states with equal public projection differ in outgoing edges then they can not be unified and an integer *mark* is introduced to distinguish them.

Algorithm 1 provides an overview of our distributed algorithm to find a solution of an MA-STRIPS problem without a need to compute complete PSMs of agents local problems. Every agent α starts with an empty PSM Γ_α which contains only the initial state of Π^α. In every iteration, every agent generates a new plan of its local problem Π^α and it adds this plan to Γ_α. Then public projection $\Gamma_\alpha^{\text{pub}}$ is computed and exchanged with other agents. This process continues until the intersection $\bigcap_\beta(\Gamma_\beta^{\text{pub}})$ is not empty.

By a new plan in the first step we mean a plan that was not generated in any of the previous iterations. To achieve this we have modified an existing planner Fast Downward[3] so that it is able to generate a plan which differs from plans provided as an input. This extension is inspired by diverse planning with homotopy class constraints [1]. Homotopy classes of plans are naturally defined by their public projections, that is, two plans belong to same homotopy class iff they have equal public projection.

In the last step of the loop, other agents public projections are incorporated into the local planning problem Π^α using the principle of *prioritizing actions* and *soft-landmarks*. Prioritizing actions are implemented using action costs so that internal actions are preferred to public actions, and α's public actions are preferred to other agent actions. When agent α finds a new local solution it sends its public projection to all the other agents. Other agents then extend their local tasks Π^β to contain duplicated actions from the received plan. These landmark actions have significantly decreased cost and they are interlinked using fresh facts to ensure they are used in the order suggested by the public plan. See [6] for details.

[3] http://www.fast-downward.org/

Algorithm 1: Distributed algorithm to find a solution an MA-STRIPS problem Π.

```
Function PsmPlanDistributed(Π^α) is
    Γ_α ← empty PSM containing only the initial state of Π^α;
    loop
        generate a new plan π_α of Π^α and extend Γ_α with π_α;
        compute public projection Γ_α^pub of Γ_α;
        exchange public projections of PSMs with other agents;
        if the intersection ⋂_β(Γ_β^pub) is not empty then
            return the intersection;
        end
        incorporate other agents Γ_β^pub into local Π^α (optional);
    end
end
```

4 Experiments and Conclusion

We have performed extensive experimental evaluation on as much standard benchmark problems as possible. For that purpose, we have converted six domains from IPC benchmarks to multi-agent. We compare our results with two state-of-the-art multi-agent planners wherever possible. Our results show best performance in *satellites* domain. In *rovers* domain it beats MA-BFS while it solves same number of problems as FMAP algorithm.

We have proposed a formal and algorithmic definition of a novel multi-agent planning scheme based on iterative building and intersecting of Finite State Machines implicitly representing local plans of the agents. The building process uses FastDownward planner for generation of the local plans, which then extends the agents' plan generative PSMs. The scheme was implemented and experimentally evaluated in various planning domains and problems for MA-STRIPS planning. The results show that in some domains it performs significantly better than state-of-the-art multi-agent planners.

This research was supported by the Czech Science Foundation (grant no. 13-22125S) and in part by a Technion fellowship.

REFERENCES

[1] Subhrajit Bhattacharya, Vijay Kumar, and Maxim Likhachev, 'Search-based path planning with homotopy class constraints.', in *SOCS*, eds., Ariel Felner and Nathan R. Sturtevant. AAAI Press, (2010).

[2] Daniel Borrajo, 'Plan sharing for multi-agent planning', in *Proc. of DMAP Workshop of ICAPS'13*, pp. 57–65, (2013).

[3] R.I. Brafman and Carmel Domshlak, 'From One to Many: Planning for Loosely Coupled Multi-Agent Systems', in *Proceedings of ICAPS'08*, volume 8, pp. 28–35, (2008).

[4] Eric Fabre, Loïg Jezequel, Patrik Haslum, and Sylvie Thiébaux, 'Cost-optimal factored planning: Promises and pitfalls', in *ICAPS*, eds., Ronen I. Brafman, Hector Geffner, Jörg Hoffmann, and Henry A. Kautz, pp. 65–72. AAAI, (2010).

[5] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman, *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006.

[6] Jan Tožička, Jan Jakubův, Karel Durkota, Antonín Komenda, and Michal Pěchouček, 'Multiagent Planning Supported by Plan Diversity Metrics and Landmark Actions', in *Proceedings ICAART'14*, (to appear).