Proceedings of the Twenty-Sixth International Conference on
Automated Planning and Scheduling (ICAPS 2016)

Online Macro Generation for Privacy Preserving Planning

Shlomi Maliah and **Guy Shani**
Information Systems Engineering
Ben Gurion University
shlomima/shanigu@bgu.ac.il

Ronen I. Brafman
Computer Science
Ben Gurion University
brafman@cs.bgu.ac.il

Abstract

Agents that use Multi-Agent Forward Search (MAFS) to do privacy-preserving planning, often repeatedly develop similar paths. We describe a simple technique for online macro generation allowing agents to reuse successful previous action sequences. By focusing on specific sequences that end with a single public action only, we are able to address the utility problem – our technique has negligible cost, yet provides both speedups and reduced communication in domains where agents have a reasonable amount of private actions. We describe two variants of our approach, both with attractive privacy preserving properties, and demonstrate the value of macros empirically. We also show that one variant is equivalent to secure MAFS.

1 Introduction

In various settings, agents may wish to cooperate to achieve joint goals, while concealing certain private facts. For example, different manufacturers may want to collaborate in the production of a good without disclosing their entire supply-chain, inventory levels, and local processes. Similarly, the army may want to outsource its food supply to external caterers without revealing the location of its bases and the number of soliders serving in each, and the caterer may wish to keep private the number of trucks it operates.

An attractive framework for such planning problems is privacy preserving planning (Nissim and Brafman 2014) which has gained increasing attention in recent years. The latest CoDMAP competition (Štolba, Komenda, and Kovacs 2015) accepted numerous submissions from 10 different groups. While several approaches were suggested, heuristic search algorithms (Maliah, Shani, and Stern 2015; Štolba and Komenda 2014; Štolba, Fišer, and Komenda 2015; Maliah, Shani, and Stern 2014) seem to be the best performers. In particular, the Multi-Agent Forward Search algorithm (MAFS) (Nissim and Brafman 2012), combined with strong heuristic estimates (Štolba, Fišer, and Komenda 2015) is perhaps the best performing.

Macro actions (macro operators) are a well known technique for speeding up heuristic search (Koedinger and Anderson 1990; Botea et al. , e.g.). Macros can be identified offline, prior to searching, or online, while searching for satisfying plans (Coles and Smith 2007; Chrpa, Vallati, and McCluskey 2015). In online macro generation, sequences of actions that are deemed useful during the search are stored to be reused when needed, potentially reducing the number of expansions, at the cost of an increased branching factor.

In MAFS, each agent plans independently in its own private search space, until a public action has been invoked. Then, the agent notifies all other agents about the result of the public action, allowing them to build upon the effects of that action. Thus, the search tree of each agent is composed of sequences of private actions, intervened by public actions of the planning agent or other agents. Often, such action sequences are heuristically preferred, causing agents to expand identical private action sequences in similar contexts.

In this short paper we use these sequences of private actions between pairs of public actions to define macro-actions, providing agents with shortcuts in the search space.

By focusing on *grounded* sequences ending with a single public action that lead to a state with an *improved* heuristic value, we are able to address the utility problem (Minton 1988): the class of macros generated is restricted, yet useful – the application of a useful public action typically requires a sequence of private actions, and this combination is likely to be required again. Moreover, macro generation is guided by the heuristic function which determines which sequence are encountered online, and filters out sequences that do not improve the heuristic value.

Our macros can be used locally, by the agent that identified them, or globally, by all agents. In the latter case, the agent must send the macro to other agents, and this may appear to disclose private information. However, it also allows other agents to send fewer states, thus hiding their internal search space, providing an efficient implementation of the recent *secure* MAFS algorithm (Brafman 2015).

We compare the two approaches for using macros and discuss their advantages and disadvantages and provide an extensive set of experiments, showing how Macro-MAFS operates better than MAFS on more complex benchmarks.

2 Background

We begin by providing background on privacy preserving collaborative planning, the MAFS algorithm, and on macro actions in classical planning.

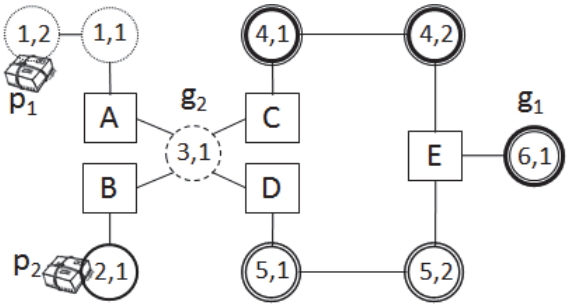

Figure 1: A logistics example, where trucks deliver packages between logistics centers, denoted by squares. Each agent covers a set of cities, denoted by circles, and labeled i, j where i is the agent covering the city and j is the local city index. The logistic centers can be entered by several agents, serving as collaboration sites.

2.1 Privacy Preserving Planning

A MA-STRIPS problem(Brafman and Domshlak 2013) is represented by a tuple $\langle P, \{A_i\}_{i=1}^{k}, I, G\rangle$ where:

- k is the number of agents.
- P is a finite set of primitive propositions (facts).
- A_i is the set of actions agent i can perform.
- I is the start state.
- G is the goal condition.

Each action $a = \langle pre(a), \mathit{eff}(a)\rangle$ is defined by its preconditions ($pre(a)$), and effects ($\mathit{eff}(a)$). Preconditions and effects are conjunctions of primitive propositions and literals, respectively. A state is a truth assignment over P. G is a conjunction of facts. $a(s)$ denotes the result of applying action a to state s. A *plan* $\pi = (a_1, \ldots, a_k)$ is a solution to a planning task iff $G \subseteq a_k(\ldots(a_1(I)\ldots)$.

Privacy-preserving MA-STRIPS extends MA-STRIPS by defining sets of variables and actions as private, known only to a single agent. $private_i(P)$ and $private_i(A)$ denote the variables and actions, respectively, that are private to agent i. $public(P)$ is the set of public facts in P. $public_i(A)$, the complement of $private_i(A)$ w.r.t. A_i is the set of public actions of agent i. Some preconditions and effects of public actions may be private, and the action obtained by removing these private elements is called its *public projection*. For ease of exposition we assume that all goals are public.

Figure 2.1 illustrates a simple logistic example in which the agents are trucks tasked with delivering packages. The set of facts P represents the location of two packages and six trucks. Each truck has three actions: move, load, and unload, corresponding to moving between locations, loading a package and unloading it. Trucks can only drive along the edges in Figure 2.1. Agents are heterogeneous and their range is restricted, such that location i, j can only be reached using truck i. The rectangles are logistic centers visited by multiple trucks that load or unload packages.

Trucks are owned by different companies that do not want to share their locations and coverage (which locations it can reach) with other companies. Thus, all the facts representing the location of trucks are private, while the facts representing whether a package is at a logistic center are public. Only the load/unload actions at the logistic centers are public, whereas the move actions are private for each agent, as well as loading and unloading at private locations.

2.2 MAFS and Secure MAFS

Multi-agent forward search (MAFS) (Nissim and Brafman 2014) is a distributed algorithm schema for forward-search planning that also preserves privacy. In MAFS, each agent maintains a separate search space with its own *open* and *closed* lists. It expands the state with the minimal heuristic value in its open list using *its own actions only*. This means that two agents (with different action sets) expanding the same state will generate *different* successor states. As no agent expands all relevant search nodes, messages must be sent between agents, informing one agent of open search nodes expanded by other agents. Thus, agents that generate a state using a public action send it to all other agents. These agents, in turn, add this state to their open list.

The MAFS schema can be applied to all search algorithms – in fact, agents can use different search algorithms and different heuristic functions, although care is needed when identifying solutions, especially if an optimal one is sought.

One of the key open issues in research on multi-agent planning is that of quantifying the level of privacy afforded by each algorithm. (Brafman 2015) distinguishes between *weak privacy*, where agents never explicitly communicate private information, and *strong privacy*, where agents can never infer private information. But there is a clear need for intermediate notions between weak and strong privacy and proof techinques for strong privacy.

To maintain agent privacy, agents using MAFS must encrypt their private state and use a privacy preserving heuristic function, such as one that is based on the public projection of existing actions. Since agents are not influenced by the private state of another agent, nor do they influence it, they can simply copy the encrypted value of the private state of other agents when they apply their own actions. With these restrictions, MAFS is weakly private.

Secure MAFS (SMAFS) (Brafman 2015) is a variant of MAFS that is provably strongly private in some limited settings. In SMAFS agent i will send a state s only if it never sent a state s' earlier, such that s and s' assign the same values to all variables that are not private to i. This implies that no agent will see two private states associated with the same non-private state of agent i. To preserve completeness, agent i must simulate the effect of the actions of other agents on states that were not sent. By recording the transformations other agents apply to a state s, the agent can emulate it on any state s' that differs from s only in the value of its private variables. These transformations do not impact its private statem and, as we show later, they can be captured naturally using macros.

2.3 Macro Actions

Macro actions have been widely studied in automated planning. A macro action is a sequence of actions that was identified as useful, typically appearing in many branches of the

search tree. A macro action is represented like any other action, using its preconditions and effects. The preconditions of the macro action are all the preconditions of actions in the sequence that are not achieved by earlier actions in the sequence. Its effects are the effects of actions in the sequence that are not destroyed by later actions in the sequence.

Macro actions can speed up search by moving deeper into the search tree with a single decision. However, each added macro increases the branching factor of the search tree. Hence, adding too many macros may increase planning time. This is known as the *utility* problem (Minton 1988).

Many existing macro learning methods such as MacroFF (Botea et al.), Wizard (Newton et al. 2007), and MUM (Chrpa, Vallati, and McCluskey 2014) use offline learning. That is, examples of problems and their solutions are analyzed to discover useful action sequences that are good candidates for macros.

Other methods, including OMA (Chrpa, Vallati, and McCluskey 2015), MacroPlanner (Jonsson 2007), and CAP (Asai and Fukunaga 2015), are semi-online. They analyze the given domain and problem instance before search commences, without using knowledge from similar domains or problems, but also without utlizing information from the search. For example, MacroPlanner and CAP generate and solve subproblems that contain fewer variables. Then, planning commences with the richer operator set.

Marvin (Coles and Smith 2007) is the only truely online macro-learning system we are aware of. It addresses a specific issue encountered by enforced hill climbing planners: overcoming plateaus and local minima. It caches useful action sequences that helped it escape from them in the past in the hope that they will be useful when future plateaus and local minima are encountered. While our method is not geared to a particular search method, it is focused on exploiting the particular features of multi-agent forward search and the exchangeability of private action sequences executed by different agents. And whereas most of the above methods seek generally applicable macros, trying to lift them, our macros are fully grounded, thus restricting their applicability, and controlling the branching factor.

Algorithm 1: MAFS for agent i

```
1  MAFS(i)
2      Insert I into open list
3      while solution not found do
4          foreach message m call process-message(m)
5          s ← extract-min(open list)
6          expand(s)
7  process-message(m = ⟨s, g_j(s), h_j(s)⟩)
8      if s is not in open or closed list ∨ g_i(s) > g_j(s)
       then
9          add s to open list and calculate h_i(s)
10         g_i(s) ← g_j(s), h_i(s) ← max(h_i(s), h_j(s))
11 alg:expand
12 expand(s)
13     move s to closed list
14     if s is a goal state then
15         broadcast s to all agents
16         if s has been broadcasted by all agents then
17             return s as the solution
18     if the last action leading to s was public then
19         send s to all relevant agents
20     apply i's successor operators to s
21     foreach successors s' do
22         if s' is not in closed list or it's heuristic value
           has improved then
23             add s' to open list
```

3 Online Macro Generation in MAFS

We generate macros online during the search for a solution plan, adding them to our action set, as we go. We first explain macro identification, and then consider two variants.

In MAFS, each agent progresses its own private search by executing private actions, but the collaborative search process is expedited when agents execute public actions. Indeed, branches in the search tree consist of a sequence of subsequences consisting of zero or more private actions followed by a public action, all by the same agent. These action "blocks" make excellent candidates for macro actions, to be reused later in the search.

Macro reuse can be very helpful in our settings. Consider for example our running logistics problem (Figure 2.1). The planning process begins with agent 1 bringing package p_1 to location A. Concurrently, agent 2 brings package p_2 to location B. Thus, agents 1,2,3 (as well as the other agents) are aware of two different public states, one where $at(p_1, A)$ and the p_2 is at its initial location, and one where $at(p_2, B)$ and p_1 is at its initial location. Agent 1 must now start searching from the latter ($at(p_2, B)$), planning again to bring p_1 to A. If agent 1 could reuse a macro for bringing p_1 to A, it could immediately publish a new public state where both $at(p_1, A)$ and $at(p_2, B)$ hold. Then, agent 3 can start planning from a state where both packages are within its reach.

Thus, identifying macros is simple — whenever an agent executes a public action a_{pub}, it generates a macro. This macro contains all private actions that were executed immediately prior to a_{pub} followed by a_{pub} itself and is saved only if the heuristic value of the current (end) state is smaller than the heuristic value of the state before the first private action was executed. We make no attempt to lift this macro.

We consider two methods for using the identified macros. In the *"private"* variant, a macro is known only to the agent that identified it, and it is added to its list of actions. The public projection of the macro is identical to the final public action in the macro. In the *"published"* variant, the macro generated is sent to other agents, allowing them, in essence, to apply public actions of the generating agent.

"Private" macros do not impact privacy, as each macro serves as a shortcut to a sequence of expansions that would be carried out anyways. Moreover, as a macro contains a single public action only at its end, it does not alter the messages sent. The preconditions of the macro are all the public preconditions of actions in the macro that are not supplied

by an earlier action, and the entire private state of the agent. The latter choice reduces the generality of the macro, but also the effective branching factor. "Public" macros are generated similarly, but to maintain privacy, the agent replaces its private state in the preconditions with an encrypted private state value. This macro is sent to all other agents. To farther generalize, an agent can treat macros received as private actions for the purpose of its own macro generation. This allows for the generation of longer macros.

At a first glance, the published macros appear to carry some privacy cost, revealing information about parent-child relationship in the process. However, they also allows other agents to send less information because, sometimes, instead of sending a state to another agent, the agent can simply apply that other agent's macros directly.

In fact, with minor modifications, we can use the "published" variant to implement SMAFS. The key property of SMAFS is that agents never send two states that differ only in the value of their private variables. Macros make it easy to achieve this property while maintaining completeness. This requires modifying the algorithm as follows: 1. Each agent maintains the list of states sent so far. Whenever MAFS requires the agent to send a state s, it will *not* send s if some state s' sent earlier is identical to s, except, possibly, for the agent's private state. 2. States not sent for this reason must be kept in a special closed list. 3. Whenever an agent receives a new macro from another agent, it must apply that macro to every state in its special closed list (in addition to adding it to its action list). 4. For completeness, macros must be generated even if they do not lead to a state with an improved heuristic value. However, it is sufficient to apply such macros only in case 3 above.

Technically, the only difference between this implementation of SMAFS and (Brafman 2015) is that in the latter agents must essentially identify the two end states of the macro – the state in which the action sequence is applied, and the resulting state, and copy the transformation to similar states. The macro that we send is slightly more general (as it refers only to the sending agent's private state), but in principle, the same macro could be deduced by an agent from the information obtained in the original version. Moreover, the proofs are not affected by these differences.

4 Empirical Evaluation

We conduct an empirical analysis of our macro variants. We experiment with a set of benchmarks from the CoDMAP competition (Štolba, Komenda, and Kovacs 2015), and two more complicated domains — MA-Blocks and MA-Logistics, where a larger number of private actions need to be executed between two consecutive public actions, and agents choose between several paths for achieving goals.

Table 1 compares MAFS without macros (denoted None), MAFS with private macros, and MAFS with published macros. The algorithms are compared on the number of actions in the final plan, runtime, the number of sent messages, and the number of state expansions.

MAFS without macros produces slightly shorter plans. As we use a best first search mechanism, macros may be used where shorter sequences of actions might do just as well. This is also reflected in the number of state expansions, which is largest for MAFS without macros. Not using macros also results in more planning time (except for the easiest domain — logistics). As expected, when publishing macros, agents can imitate the search process of other agents, avoiding the need for other agents to publish the revised states. Thus, the number of messages when publishing macros is lower on almost all domains.

Domain	None	Private	Published
	Actions		
MA-Blocks	**28.1**	28.2	28.2
MA-Logistics	**236**	236.3	242.5
rover	**42.5**	48.3	47.2
logistics	**50**	**50**	52.8
elevator	**56**	**56**	60.7
zenotravel	**26.8**	29.3	32.8
	State expansions		
MA-Blocks	99.8	**72.2**	82.7
MA-Logistics	3512.4	**2345.5**	2546.3
rover	425.8	**301.3**	439.6
logistics	205.7	**203.5**	212.7
elevator	761.1	753.2	**751.5**
zenotravel	171.5	163.1	**153.1**
	Time (secs)		
MA-Blocks	467.9	**297.4**	364.2
MA-Logisitcs	90.7	**59.3**	83.6
rover	27.3	**19.8**	30.2
logistics	**2.7**	2.8	3.9
elevetor	48.6	**48.5**	47.6
zenotravel	32.3	27.4	**17.8**
	Messages		
MA-Blocks	258.8	244.9	**202.7**
MA-Logisitcs	6072.8	**4829.5**	5435.5
rover	859.1	882.9	**807.4**
logistics	1867.1	1852.7	**1825.1**
elevetor	3145.4	3145.5	**2989.2**
zenotravel	957.8	1097.6	**727.5**

Table 1: Comparing the performance of macro variants.

	Private			Published		
Domain	Count	Length	Portion	Count	Length	Portion
MA-Blocks	3.76	2.038	0.26	5.04	2.29	0.32
MA-Logistics	16.26	4.09	0.44	15.51	3.70	0.40
logistics	3.8	2.23	0.17	6.9	2.11	0.25
elevators	4	2.28	0.16	8.34	2.20	0.33
rovers	2.85	2.18	0.15	7.75	2.28	0.38
zenotravel	2.3	1.7	0.14	5.1	2.20	0.29

Table 2: Macro use in the solution plan

Table 2 sheds some light on macro usage. We report how many macros appear on average in the solution plan, their average length, and the portion of actions in the plan that originates from macros. While the lengths of the macro are almost identical for both variants, more macro actions were used in the final plan when publishing macros. This may be because when multiple agents can use a macro, it is more likely that one of them will insert it into the final plan.

Table 3 shows the impact of macros on the search: how

Domain	Generated	Applied	Applied per state
	Private Macros		
MA-Blocks	15	10	0.13
MA-Logistics	216.78	351.825	0.16
logistics	55.25	22.26	0.11
elevator	72.8	61.27	0.081
rover	42.64	28.28	0.094
zenotravel	32.77	20.88	0.13
	Public Macros		
MA-Blocks	77 / 107	15.13	0.18
MA-Logistics	975 / 7321	849.19	0.33
logistics	468 / 2567	86.65	0.41
elevator	203 / 4829	84.45	0.11
rover	149 / 968	486.71	1.1
zenotravel	72 / 648	26.56	0.17

Table 3: Impact of published macros on search

many macros were generated, applied in total, and applied per state. For published macros the generated macros distinguish between macros added to the action set, and macros required to maintain completeness (see Section 3). We see that in the case of private macros, the impact on the branching factor is minimal. Even for published macros, the increase in branching factor is almost always less than half, except for Rovers.

5 Conclusion

We show how macro actions can be identified and used in the MAFS algorithm. We demonstrate that macros speed up MAFS, reducing the number of sent messages. We explain how identifying and publishing macros preserves privacy, resulting in privacy guarantees equivalent to Secure MAFS.

In the future we intend to study macros in the context of other privacy preserving algorithms such as GPPP(Maliah, Shani, and Stern 2014), as well as study additional methods for identifying useful macros. Another interesting direction is to explore the combinations of macros and pruning techniques.

Acknowledgments: We thank the reviewers for their useful comments. This work was supported by ISF Grant 933/13, by the Helmsley Charitable Trust through the Agricultural, Biological and Cognitive Robotics Center of Ben-Gurion University of the Negev, and by the Lynn and William Frankel Center for Computer Science.

References

Asai, M., and Fukunaga, A. 2015. Solving large-scale planning problems by decomposition and macro generation. In *ICAPS15*.

Botea, A.; Enzenberger, M.; Müller, M.; and Schaeffer, J. Macro-ff: Improving AI planning with automatically learned macro-operators. *J. Artif. Intell. Res. (JAIR)* 581–621.

Brafman, R. I., and Domshlak, C. 2013. On the complexity of planning for agent teams and its implications for single agent planning. *Artificial Intelligence* 198:52–71.

Brafman, R. I. 2015. A privacy preserving algorithm for multi-agent planning and search. In *the International Joint Conference on Artificial Intelligence (IJCAI)*, 1530–1536.

Chrpa, L.; Vallati, M.; and McCluskey, T. L. 2014. MUM: A technique for maximising the utility of macro-operators by constrained generation and use. In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2014, Portsmouth, New Hampshire, USA, June 21-26, 2014*.

Chrpa, L.; Vallati, M.; and McCluskey, T. L. 2015. On the online generation of effective macro-operators. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, 1544–1550.

Coles, A., and Smith, A. 2007. Marvin: A heuristic search planner with online macro-action learning. *J. Artif. Intell. Res. (JAIR)* 28:119–156.

Jonsson, A. 2007. The role of macros in tractable planning over causal graphs. In *IJCAI'07*, 19361941.

Koedinger, K. R., and Anderson, J. R. 1990. Abstract planning and perceptual chunks: Elements of expertise in geometry. *Cognitive Science* 14(4):511–550.

Maliah, S.; Shani, G.; and Stern, R. 2014. Privacy preserving landmark detection. In *the European Conference on Artificial Intelligence (ECAI)*, 597–602.

Maliah, S.; Shani, G.; and Stern, R. 2015. Privacy preserving pattern databases. In *ICAPS workshop on Distributed and Multi-Agent Planning (DMAP)*.

Minton, S. 1988. Quantitative results concerning the utility of explanation-based learning. In *Proceedings of the 7th National Conference on Artificial Intelligence. St. Paul, MN, August 21-26, 1988.*, 564–569.

Newton, M. A. H.; Levine, J.; Fox, M.; and Long, D. 2007. Learning macro-actions for arbitrary planners and domains. In *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling, ICAPS 2007, Providence, Rhode Island, USA, September 22-26, 2007*, 256–263.

Nissim, R., and Brafman, R. I. 2012. Multi-agent A* for parallel and distributed systems. In *AAMAS*, 1265–1266.

Nissim, R., and Brafman, R. I. 2014. Distributed heuristic forward search for multi-agent planning. *Journal of Artificial Intelligence Research (JAIR)* 51:293–332.

Štolba, M., and Komenda, A. 2014. Relaxation heuristics for multiagent planning. In *International Conference on Automated Planning and Scheduling (ICAPS)*.

Štolba, M.; Fišer, D.; and Komenda, A. 2015. Admissible landmark heuristic for multi-agent planning. In *International Conference on Automated Planning and Scheduling (ICAPS)*.

Štolba, M.; Komenda, A.; and Kovacs, D. L. 2015. Competition of distributed and multiagent planners (codmap). *The International Planning Competition (WIPC-15)* 24.