Proceedings of the Twelfth International
Symposium on Combinatorial Search (SoCS 2019)

Novelty Messages Filtering for Multi Agent Privacy-Preserving Planning

Alfonso E. Gerevini
Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Brescia, Italy

Nir Lipovetzky
School of Computing and Information Systems
The University of Melbourne

Nico Peli, Francesco Percassi, Alessandro Saetti, Ivan Serina*
Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Brescia, Italy

Abstract

In multi-agent planning, agents jointly compute a plan that achieves mutual goals, keeping certain information private to the individual agents. Agents' coordination is achieved through the transmission of messages. These messages can be a source of privacy leakage as they can permit a malicious agent to collect information about other agents' actions and search states. In this paper, we investigate the usage of novelty techniques in the context of (decentralised) multi-agent privacy-preserving planning, addressing the challenges related to the agents' privacy and performance. In particular, we show that the use of novelty based techniques can significantly reduce the number of messages transmitted among agents, better preserving their privacy and improving their performance. An experimental study analyses the effectiveness of our techniques and compares them with the state-of-the-art. Finally, we evaluate the robustness of our approach, considering different delays in the transmission of messages as they would occur in overloaded networks, due for example to massive attacks or critical situations.

Introduction

Several frameworks for decentralised multi-agent (DMA) planning have been formalized and developed in the last few years, e.g., (Brafman and Domshlak 2008; Nissim and Brafman 2014; Torreño, Onaindia, and Sapena 2014). An important issue of these approaches is related to how they handle agents' privacy; in fact, for DMA planning agents have private knowledge that they do not want to share with others during the planning process and plan execution. This issue prevents the straightforward usage of most of the modern techniques developed for centralized (classical) planning, which are based on heuristic functions computed by using the knowledge of all the involved agents.

In DMA planning, computing search heuristics using the knowledge of all the involved agents may require many exchanges of knowledge among agents, and this may compromise the agents' privacy. For preserving the privacy of the involved agents, the distance from a search state to the goal states can be estimated by using either the knowledge of a *single* agent alone, or the public projections of actions of the involved agents. However, this estimate is much more inaccurate than using the knowledge of all the agents.

Given that for classical planning best-first width search performs very well even when the estimate of the goal distance is inaccurate (Lipovetzky and Geffner 2012), such a procedure represents a good candidate to effectively solve MA-planning problems without compromising the agents' privacy. Recently, for MA planning Gerevini et al. (2019) proposed an effective search procedure called MA-BFWS that uses width-based exploration in the form of novelty-based preferences to provide a complement to goal-directed heuristic search. In order to preserve the privacy of the involved agents, the private knowledge shared among agents is encrypted. An agent α_i shares with the other agents a description of every reached search state in which all the private facts of α_i that are true in a state are substituted with a string obtained by encrypting all the private fact names of α_i together. Notably, this encryption has an impact on the measure of novelty, and hence also affects the definition of the heuristic guiding the search (Gerevini et al. 2019).

In this paper we investigate the use of novelty to filter the messages sent by each agent, we propose different methods to exploit such filtering within forward search MA planning, and we discuss its properties in terms of privacy. The following sections present the background on width-based search and privacy in MA planning, discuss related work, propose our filtering techniques based on the notion of novelty, show the results of an experimental study to evaluate the effectiveness of the proposed novelty message heuristics, and finally give the conclusions and mention future work.

Background

In this section, first we introduce the MA-STRIPS planning problem, then we describe some prominent width-based search procedures developed for classical planning.

The MA-STRIPS planning problem

Our work relies on MA-STRIPS, a "minimalistic" extension of the STRIPS language for MA planning (Brafman and Domshlak 2008), that is the basis of the most popular definition of MA-planning problem (see, e.g., (Nissim and Brafman 2014; Brafman 2015; Maliah, Shani, and Stern 2014; Maliah, Stern, and Shani 2016; Nissim and Brafman 2012)).

*Corresponding author. Email: ivan.serina@unibs.it

Definition 1 *A* MA-STRIPS *planning problem* Π *for a set of agents* $\Sigma = \{\alpha_i\}_{i=1}^n$ *is a 4-tuple* $\langle \{A_i\}_{i=1}^n, P, I, G\rangle$ *where:*

- A_i *is the set of actions agent* α_i *is capable of executing, and s.t. for every pair of agents* α_i *and* α_j $A_i \cap A_j = \emptyset$*;*
- P *is a finite set of propositions;*
- $I \subseteq P$ *is the initial state;*
- $G \subseteq P$ *is the set of goals.*

Each action a consists in a name, a set of preconditions, $Prec(a)$, representing facts required to be true for the execution of the action, a set of additive effects, $Add(a)$, representing facts that the action makes true, a set of deleting effects, $Del(a)$, representing facts that the action makes false, and a real number, $Cost(a)$, representing the cost of the action. A fact is *private* for an agent if other agents can neither achieve, destroy nor require the fact (Brafman and Domshlak 2008). A fact is *public* otherwise. An action is *private* if all its preconditions and effects are private; the action is *public*, otherwise. A state obtained by executing a public action is said to be public; otherwise, it is private. In the rest of the paper, we denote by $public(s)$ the public part of a state s.

To maintain agents' privacy, the private knowledge shared among agents can be encrypted. An agent can share with the other agents a description of its search state in which each private fact that is true in a state is substituted with a string obtained by encrypting the fact name (Bonisoli et al. 2018). This encryption of states does not reveal the names of the private facts of each agent α_i to other agents, but an agent can realize the existence of a private fact of agent α_i and monitor its truth value during search. This allows the other agents to infer the existence of private actions of α_i, as well as to infer their causal effects. Another way of sharing states containing private knowledge during the search is to substitute, for each agent α_i, all the private facts of α_i that are true in a state with a string obtained by encrypting all private fact names of α_i together (Nissim and Brafman 2014). Such a string denotes a dummy private fact of α_i, which is treated by other agents as a regular fact. The work presented in this paper uses this latter method for the encryption of states. With this method, other agents can only infer the existence of a group of private facts of α_i, since the encrypted string contained in the states exchanged by α_i substitutes a group of an arbitrary number of private facts of α_i.

Privacy

Privacy in MA-STRIPS planning is concerned with guaranteeing that the private information of an agent α_i remains known only by agent α_i, namely: its private variables and values, as well as the existence, structure and cost of its private actions. Brafman (2015) defines an algorithm as *weakly privacy* preserving if no value of private variables is shared with other agents, and the information shared among agents consists of *public projections* of public actions, i.e. private preconditions and effects are dropped from public actions. Algorithms for MA planning typically achieve weak privacy by encrypting the private variables of a state before sending it.

Weakly private algorithms allow agents to track changes in states sent by one agent and deduce the existence of private variables. In contrast, an algorithm is *strongly private* if no agent α_i can deduce about the existence of i) a value or variable private to agent α_j, $i \neq j$, and ii) the model of private actions of α_j, beyond 1) what its own actions A_i reveals, 2) the public projection of actions A_j, and 3) the public projection of the actions in the solution plan.

One way to hinder the deduction ability of an agent is randomizing the order of exchanged messages, which can be achieved by delaying messages with random times from a given probabilistic distribution. This has not been studied formally, but we experimentally evaluate the impact of delays in MA planning algorithms.

Brafman (2015) also proposes *secure-MAFS*, a complete and sound forward search algorithm that achieves *strong privacy* when the heuristic used is independent from the private part of the problem and all actions have unary cost. The key insight relies on making sure that an agent sends a state s to other agents iff the public projection of state s has never been sent before. This ensures that agents never receive two states with the same public projection from the same source agent, which is sufficient to guarantee that agents cannot distinguish between the execution of *secure-MAFS* among $\Pi, \Pi' \in C$, where C is an equivalence class containing all problems that share the same public solution space as the original problem Π being solved (Brafman 2015; Tožička, Štolba, and Komenda 2017). Other notions of privacy have been proposed; e.g., cardinality privacy prevents an agent from inferring the number of private objects of the same type managed by other agents (Maliah, Shani, and Stern 2016).

Width-based Search

Pure width-based search algorithms are exploration algorithms that do not pay attention to the problem goals at all. The simplest of such algorithm is $IW(1)$, which is a plain breadth-first search where newly generated states that do not make an atom $X = x$ true for the first time in the search are pruned. The algorithm $IW(2)$ is similar except that a state s is pruned when there are no atoms $X = x$ and $Y = y$ such that the pair of atoms $X = x, Y = y$ is true in s and false in all the states generated before s.

$IW(k)$ is a standard breadth-first search except that newly generated states s are pruned when their "novelty" is greater than k, where the novelty of s is i iff there is a tuple t of i atoms such that s is the first state in the search that makes all the atoms in t true, with no tuple of smaller size having this property (Lipovetzky and Geffner 2012). While simple, it has been shown that $IW(k)$ manages to solve arbitrary instances of many of the standard benchmark domains in low polynomial time provided that the goal is a single atom. Such domains can be shown to have a small and bounded *width* w that does not depend on the instance size, which implies that they can be solved (optimally) by running $IW(w)$. Moreover, $IW(k)$ runs in time and space that are exponential in k and not in the number of problem variables.

The procedure IW, that calls $IW(1)$ and $IW(2)$, sequentially, has been used to solve instances featuring mul-

tiple (conjunctive) atomic goals, in the context of Serialized IW (SIW) (Lipovetzky and Geffner 2012), an algorithm that calls IW for achieving one atomic goal at a time. In other words, the j-th subcall of SIW stops when IW generates a state s_j that consistently achieves j goals of G. The state s_j *consistently achieves* $G_j \subseteq G$ if s_j achieves G_j, and G_j does not need to be undone in order to achieve G. This last condition is checked by testing whether $h_{max}(s_j) = \infty$ is true once the actions that delete atoms from G_j are excluded. While SIW is an incomplete blind search procedure (if dead-ends exist), it turns out to perform better than a greedy best-first search guided by standard delete relaxation heuristics (Lipovetzky and Geffner 2012).

Width-based exploration in the form of simple novelty-based preferences instead of pruning can provide an effective complement to goal-directed heuristic search without losing completeness. Indeed, it has been recently shown that for classical planning the combination of width-based search and goal-directed heuristic search, called best-first width search (BFWS), yields a search scheme that is better than both, and outperforms the state-of-the-art planners (Lipovetzky and Geffner 2017).

BFWS(f) with $f = \langle h, h_1, ..., h_n \rangle$ is a standard best-first search that uses the function h to rank the nodes in the *open list*, breaking ties lexicographically with n functions $h_1, \ldots, h_n$. The primary evaluation function h is given by the novelty measure of the node. To integrate novelty with goal directed heuristics, the notion of novelty used by BFWS is different from that used for the breadth-first search IW. For BFWS, given the functions $h_1, \ldots, h_n$, the novelty $w(s')$ of a newly generated state s' is i iff there is a tuple (set) of i atoms $X_i = x_i$ and no tuple of smaller size, that is true in s but false in all previously generated states s' with the same function values $h_1(s') = h_1(s), \ldots,$ and $h_n(s') = h_n(s)$. For example, a new state s has novelty 1 if there is an atom $X = x$ that is true in s and false in all the states s' generated before s where $h_i(s') = h_i(s)$ for all i. In the rest of the paper, the novelty measures w is sometimes denoted as $w_{(h_1,\ldots,h_n)}$ in order to make explicit the functions $h_1, \ldots, h_n$ used in the definition and computation of w.

Related Work

A MA-planning algorithm similar to ours is MAFS (Nissim and Brafman 2014). MAFS is a distributed best first search that for each agent considers a separate search space. The existing work investigating the use of a distributed A* for partial-order MA-planning shares our motivations (Torreño, Onaindia, and Sapena 2014). Differently from this approach, our MA-planning procedure searches in the space of world states, rather than in a space of partial plans, and it exchanges states among agents rather than partial plans.

Using width-based search for MA planning is not a novel idea. IW search was used for solving a classical planning problem obtained from the compilation of a multi-agent planning problem (Muise, Lipovetzky, and Ramirez 2015). This work is for centralized MA planning, while our work investigates the distributed MA planning problem. Bazzotti et al. (2018) study the usage of Serialized-IW (abbreviated by MA-SIW) in the setting of distributed MA planning. The MA problem solved by MA-SIW is split into a sequence of episodes, where each episode j is a subproblem solved by IW, returning a path to a state where one more problem goal has been achieved with respect to the last episode $j-1$. Finally, Gerevini et al. (2019) study the properties of MA-BFWS in terms of the width of MA-STRIPS problems, and propose several evaluation functions combining novelty and new heuristic functions where state-of-the-art performance is achieved without sharing the public projection of actions across agents. Differently from these approches using width-based search, in this work we study the usage of novelty to filter the messages sent by each agent, instead of to prune or evaluate search states.

On the use of Novelty for message transmission

The aim of our work is to reduce the number of search states exchanged among agents during the search phase. Each agent retains, therefore does not send to other agents, the search states whose novelty value called **outgoing novelty** exceeds a given threshold. The outgoing novelty is computed considering the public part of the search states previously transmitted to the other agents.

Definition 2 *The **outgoing novelty** of a state s given m functions $h_1, \ldots, h_m$ is k (denoted as $w^{out}_{h_1,\ldots,h_m}(s) = k$) iff there is a tuple (conjunction) of k atoms and no smaller tuple, that is true in the* public part *of s and false in the* public part *of all states s' previously transmitted with the same function values, i.e., with $h_i(public(s')) = h_i(public(s))$ for $1 \leq i \leq m$. If no such tuple exists, then $w^{out}_{h_1,\ldots,h_m}(s)$ is set to the maximum value $|P^{pub}| + 1$, where $|P^{pub}|$ is the number of public propositions in the problem.*

Based on the outgoing novelty value, a state s can be withheld or transmitted to other agents.

Definition 3 *The **withheld states** are the states that have not been sent to the other agents because of the fact that their outgoing novelty exceeds a given threshold.*

In order to preserve the completeness of the algorithm, these states can be transmitted to the other agents under specific conditions. To describe these conditions, it is necessary to introduce the definition of **partially empty search**.

Definition 4 *The search of an agent α is **partially empty** when the following conditions hold:*

1. *the* open list *of α is empty;*
2. *α has no other entry message to process;*
3. *α has at least one withheld state.*

If condition 1 and 2 hold but condition 3 does not hold, then the search of α is **empty**.

An agent whose search phase is empty or partially empty is called "*waiting*". When an agent α is *waiting*, it can send a part of, or even all, its withheld states (if any) to the other agents in order to "reinvigorate" the search process of the other agents, or it can ask the other agents to send their withheld states to it in order to restore its search process. In contrast, α could wait that at least a given number of agents, indicated with $num\_waiting$, are in the same condition before

sending its withheld states or asking the withheld states of the other agents. With this purpose, the agents communicate if their search is empty or partially empty to the others; when the number of waiting agents exceeds value $num\_waiting$, the agents transmit their withheld states.[1] In particular, we distinguish among three situations:

- $num\_waiting = 1$, agents transmit/request their withheld states when at least one agent is waiting;
- $num\_waiting = half$, agents transmit/request their withheld states when at least half the agents in the problem is waiting;
- $num\_waiting = all$, agents transmit/request their withheld states when all the agents in the problem is waiting.

As previously explained, the waiting agents can transmit/request withheld states to the other agents. Both these situations can heavily influence the search process. We considered three configurations, indicated with who_send, in order to define which agents send the withheld states. In particular when

- $who\_send = waiting$, the *waiting* agents send their withheld states;
- $who\_send = not\ waiting$, the *not waiting* agents send their withheld states;
- $who\_send = all$, all the agents send their withheld states.

The best configuration seems to be the second one ($who\_send = not\ waiting$), because since the waiting agents are idle and need states from the other agents to restore their search. On the other hand, with $who\_send = waiting$, the waiting agents can take advantage of the period of inactivity to send their withheld states.

When the previous conditions are verified the agents can send all their withheld states or only a part of them. Specifically, we considered four different configurations, indicated with $num\_withheld\_states$, that specifies which states, among the withheld ones, have to be sent when necessary:

- $num\_withheld\_states = none$, no state is sent (completeness is lost);
- $num\_withheld\_states = 1$, one state at the time is sent (the one with lowest heuristic value);
- $num\_withheld\_states = group$, all withheld states with the lowest heuristic value are sent;
- $num\_withheld\_states = all$, all withheld states are sent.

Privacy, Soundness and Completeness

We study the theoretical properties of novelty-based message filtering for sound and complete MA forward search planners. Without loss of generality, we focus on MA-BFWS, which is weakly privacy preserving: it does not require sharing the public projection of public actions, and sends messages containing only descriptions of states obtained by encrypting private facts.

[1]The transmission of the message "empty search space" among the agents is necessary in order to allow the agents to terminate their search before the timeout.

Theorem 1 MA-BFWS *with novelty messages filtering is sound and complete, iff* $num\_withheld\_states \neq none$.

Proof sketch. If $num\_withheld\_states \neq none$, then eventually all public states are going to be sent. $num\_waiting$ and who_send only changes the order of the exchanged messages, but does not preclude that messages will be sent later in the search. □

Following previous definitions of strong privacy (Brafman 2015; Tožička, Štolba, and Komenda 2017), given a problem Π, C is the set of problems Π' where their public projection is equivalent, i.e., $\Pi^{pub} = \Pi'^{pub}$ and the set of public projections of reachable public states for Π is the same as for Π'. Note that Π and Π' differ only in their private parts. MA-BFWS is strong privacy preserving with respect to the class C of MA problems, iff all agents cannot distinguish the execution of MA-BFWS with input Π and $\Pi' \in C$. An agent cannot distinguish iff the number of messages sent by an agent is the same for both problems, and the number of these messages does not depend on the private part of the problem (Tožička, Štolba, and Komenda 2017).

Theorem 2 MA-BFWS *is incomplete but strong privacy preserving for the problems in class C, for any threshold $k \in \{0, .., |P^{pub}|\}$ when used with:*

1. *$num\_withheld\_states = none$, namely pruning messages with outgoing novelty greater than the threshold;*
2. *no function h_i is used in Definition 2;*
3. *the heuristic functions used to guide the search are agnostic of the cost and private propositions of the agents.*

Proof. We assume MA-BFWS encrypts the private part of s and no information is leaked from the communication. By Definition 2, for any value of outgoing novelty, a state s is sent iff no other state s' with the same public projection has been sent before. The number of sent messages depends only on the public part of the problem, as the private part is not taken into account in the computation of outgoing novelty. As both Π and Π' share the same public problem and reachable public state space, the number of messages sent will be equivalent. Therefore, it is strong privacy preserving. □

The state expansion order of MA-BFWS is independent from the private part of the problem if the search is guided, for example, by $f = \langle w_{(\#g)}, d \rangle$, where $w_{(\#g)}$ is the novelty over the public projection of the states using a goal counting heuristic $\#g$, breaking ties with the depth d of the public actions leading to the current state. The version, with $num\_withheld\_states = none$, is incomplete if a state whose public projection has been sent before needs to be sent again in order to find a solution. MA-BFWS can be made complete by using a strategy similar to that proposed for secure-MAFS (Brafman 2015).

When strong privacy cannot be preserved, reducing the number of exchanged messages is a good strategy to decrease the deductive capability of other agents. This can be accomplished by filtering messages according to the notion of novelty. Indeed, this filtering makes sure that messages

Performance	$nw = all$	$nw = 1$	$nw = half$
Coverage (320)	259	279	**280**
Avg time	**5.01**	8.57	5.13
Avg quality	185.89	185.79	**184.75**
k-messages	**75.32**	543.43	76.69
k-states	144.26	308.11	**144.22**
IPC Quality	222.81	239.46	**240.31**
IPC Time	241.68	252.42	**258.06**
Stdev Time	**0.53**	2.8	1.01
Stdev Quality	**15.18**	16.72	15.19

Table 1: Performance of MA-BFWS with $num\_waiting$ (abbreviated with nw) set to 1, $half$, and all in terms of number of solved problems, average CPU time (in seconds), average plan cost, number of sent messages (in thousands), number of expanded states (in thousands), IPC time, IPC quality, standard deviation of the CPU times, and standard deviations of the plan costs. The best performances are indicated in bold.

Performance	$ws = wait$	$ws = not\ wait$	$ws = all$
Coverage (320)	279	278	**280**
Avg time	5.59	6.04	**5.5**
Avg quality	180.0	**178.61**	179.64
k-messages	**88.7**	109.72	92.79
k-states	227.6	228.48	**178.97**
IPC Quality	238.03	238.58	**240.39**
IPC Time	**259.76**	257.82	259.21
Stdev Time	**1.14**	1.73	1.27
Stdev Quality	16.88	14.86	**14.72**

Table 2: Performance of MA-BFWS with $num\_waiting = half$, and who_send (abbreviated with ws) set to $wait$, $not\,wait$, and all in terms of number of solved problems, average CPU time (in seconds), average plan cost, number of sent messages (in thousands), number of expanded states (in thousands), IPC time, IPC quality, standard deviation of the CPU times, and standard deviations of the plan costs. The best performances are indicated in bold.

with different state variable values are sent first, before sending states with repeating values that can lead to information leakage.

Experiments

In this section, we present an experimental study aimed at testing the effectiveness of the filtering techniques described so far. First, we describe the experimental setting; then we evaluate the effectiveness of our heuristics and compare the performance of our approach with the state-of-the-art; finally, we experiment different delays in the transmission of messages.

Our code is written in C++, and exploits the Nanomsg open-source library to share messages (Sustrik 2016).[2] Each agent uses three threads, two of which send and receive messages, while the other one conducts the search, so that the search is asynchronous w.r.t. the communication routines. The behaviour of MA-BFWS depends on the order with which the messages are received by an agent. Each time a run of MA-BFWS is repeated, the agents' threads can be scheduled by the operative system differently, so that the behaviour of MA-BFWS can also be different. Thereby, for each problem of our benchmark, we run MA-BFWS five times and consider the performance of the algorithm as the median over the five runs. When MA-BFWS exceeds the CPU-time limit for more than two of the five runs, we consider the problem unsolved.

In our experiments, MA-BFWS search uses MA-BFWS uses heuristic $f_6 = \langle w_{(G_\perp,\#r)}, G_\perp, \#r \rangle$, where novel search states are expanded first, breaking ties with counters based on the goals $G_\perp$ and a relaxed plan $\#r$ computed once from the initial state. $G_\perp$ counts the number of unachieved goals, and $\#r$ counts how many facts from a single relaxed have been achieved on the way to a state s. For more details the interested reader can see the work by Gerevini et al. (2019). The benchmark used in our experiments includes twelve domains proposed by Štolba, Komenda, and Kovacs (2015) for the distributed track of the first international competition on distributed and multi-agent planning (CoDMAP), and four domains MA-Blocksworld (shortly, MA-BW), MA-Blocksworld-Large (MA-BW-L), MA-Logistics (MA-Log), MA-Logistics-Large (MA-Log-L), which were proposed by Maliah, Brafman, and Shani (2017). In the following, these latter four domains are abbreviated to MBS. The difference w.r.t. the CoDMAP domains Blocksworld and Logistics is that for the domains of MBS many private actions need to be executed between two consecutive public actions, some goals can be private, and agents must choose among several paths to achieve goals.

All tests were run on an Ubuntu server with 24 cores Intel Xeon E5-2620 with 2 GHz and 128 Gbytes of RAM. Given a MA-planning problem, for each agent of the problem we limited the usage of resources to 3 CPU cores and 8 GB of RAM. Moreover, unless otherwise specified, the time limit was 5 minutes, after which the termination of all threads was forced.

Novelty filtering

The same measure of novelty $w_{(G_\perp,\#r)}$ used for guiding the search is used for filtering messages. We denote by w^{out} the maximum value of outgoing novelty a state s can have to not be withheld. We experimentally study different conditions under which an agent can decide to transmit (a part of) its withheld states. In our experiments, unless differently specified, MA-BFWS is used with $num\_waiting = half$, $who\_send = all$, $num\_withheld\_states = group$, and $w^{out} = 1$. The latter condition means that states with outgoing novelty greater than one have been withheld.

In the rest of the paper, for each experiment the average values are computed over the problems solved by all the compared approaches. The lower the average plan quality, the better the performance. Time and quality scores are measured by the score functions used for the seventh international planning competition. Higher values indicate better

[2] Our code and experimental data will be made available.

Performance	nws= $none$	nws= 1	nws= all	nws= $group$
Coverage (320)	258	272	275	**280**
Avg time	**5.05**	5.28	5.22	5.27
Avg quality	186.13	186.11	**185.27**	186.22
Avg k-messages	**79.31**	83.74	85.55	82.72
Avg k-states	**146.67**	180.31	151.72	153.73
Time score	241.47	250.44	253.54	**258.26**
Quality score	222.2	231.85	236.78	**238.51**
Stdev Time	**0.56**	1.22	1.04	0.85
Stdev Quality	15.52	16.02	**14.9**	15.15

Table 3: Performance of MA-BFWS with $num\_waiting = half$, $who\_send = all$, and $num\_withheld\_states$ (abbreviated with nws) set to 1, all, and $group$ in terms of number of solved problems, average CPU time (in seconds), plan cost, number of sent messages (in thousands), number of expanded states (in thousands), time score, quality score, standard deviation of the CPU times, and standard deviations of the plan costs. The best performances are indicated in bold.

performance.[3]

The first experiment we conducted concerns the decision about when an agent sends its withheld states. The results in Table 1 show that $num\_waiting = all$ is the configuration that sends the fewest states, but it is also the one with the far lowest coverage. This is probably due to the fact that requiring that all agents are in waiting state is a condition that reduces the transmission of withheld states too much. With $num\_waiting = 1$ many more states than with other configurations are sent, as every agent sends its $witheld$ states as soon as a single agent is waiting. With $num\_waiting = half$ MA-BFWS obtains a good trade-off. In terms of all the considered measures of performance, $num\_waiting = half$ is the best configuration except for the average number of sent messages and the average CPU time. However, for those measures the performance gap to the best configuration is quite limited.

In Table 2, we experimentally evaluate the performance of MA-BFWS for different configurations of who_send, that specifies which agents send withheld states. We can see that there is no big gap in the performance of these configurations. For the other experiments in the paper, we use $who\_send = all$ because it performs slightly better in terms of coverage and time, while the average number of exchanged states is pretty close to the best value obtained for $who\_send = waiting$.

Table 3 shows the results for different configurations of $num\_withheld\_states$. With $num\_withheld\_states = none$ the withheld states are not sent; the result is similar to the one obtained with $num\_waiting = all$ (in Table 1), confirming the fact that in that case too few states were shared. By sending one state at a time, i.e., with $num\_withheld\_states = 1$, we have an average number of exchanged messages and an average execution time very close to the configuration for which states are sent in group ($num\_withheld\_states = group$), although eight fewer problems are solved. Finally, as expected, with $num\_withheld\_states = all$ we have the highest number of exchanged messages; probably more than necessary given that the performance, in terms of CPU time and solved problems, is worse than with $num\_withheld\_states = group$. This shows that increasing the number of sent messages has a computational cost, since the average number of expanded states within the time limit used for our experiments is lower than with $num\_withheld\_states$ set to 1 or $group$.

[3]For details on the scores of the seventh planning competition, see http://www.plg.inf.uc3m.es/ipc2011-learning.

Domain	no filtering	$w^{out} = 1$	$w^{out} = 2$
From CoDMAP			
Blocksworld	**20**	**20**	**20**
Depot	**20**	18	**20**
DriverLog	**20**	**20**	**20**
Elevators	**20**	**20**	**20**
Logistics	19	**20**	**20**
Rovers	**20**	**20**	**20**
Satellites	**20**	**20**	**20**
Sokoban	14	14	**17**
Taxi	**20**	**20**	**20**
Wireless	**2**	**2**	**2**
Woodworking	9	**12**	**12**
Zenotravel	**20**	**20**	**20**
From MBS			
MA-BW	**19**	**19**	**19**
MA-BW-L	**15**	**15**	**15**
MA-Log	19	**20**	**20**
MA-Log-L	**20**	**20**	**20**

Performance	no filtering	$w^{out} = 1$	$w^{out} = 2$
From CoDMAP			
Coverage (320)	277	280	**285**
Avg time	16.32	**6.14**	9.69
Avg quality	179.79	**174.53**	181.26
Avg k-messages	1214.2	**100.6**	547.18
Avg k-states	480.18	**236.85**	323.74
Time score	228.73	**258.69**	245.27
Quality score	229.51	**240.3**	239.25
Stdev Time	3.68	**1.71**	1.97
Stdev Quality	17.66	**14.66**	17.23

Table 4: Coverage (upper table) of MA-BFWS without outgoing novelty filtering, with outgoing novelty w^{out} equal to 1 and 2 for domains of CoDMAP and MBS benchmarks; their performance (bottom table) in terms of number of solved problems, average CPU time (in seconds), plan cost, number of sent messages (in thousands), number of expanded states (in thousands), time score, quality score, standard deviation of the CPU times, and standard deviations of the plan costs. The best performances are indicated in bold.

In Table 4, we show the results of MA-BFWS using values of outgoing novelty w^{out} equal to 1 and 2 w.r.t. without filtering messages. It is important to remark that the computation and memory cost of determining that the outgoing novelty of a state is k is exponential in k, since all the tuples of size up to k but one may be stored and considered. For efficiency, we simplify the computation of the outgoing novelty to only 2 levels for $w^{out} = 1$, and only 3 levels for $w^{out} = 2$, i.e., the outgoing novelty for $w^{out} = 2$ is deter-

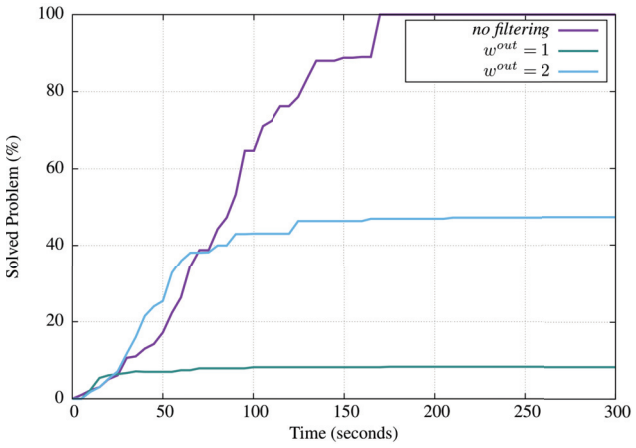

Figure 1: Percentage of messages sent by MA-BFWS without outgoing novelty filtering, MA-BFWS with outgoing novelty w^{out} equal to 1 and 2 for different CPU-time limits ranging from 0 to 300 seconds.

mined to be equal to 1, 2, or greater than 2.

The best experimented configuration is $w^{out} = 1$. The benefits of this configuration are:

- A slight improvement of the coverage w.r.t. MA-BFWS without message filtering.
- The overall number of exchanged messages is drastically reduced, by 91%. As previously noted, this reduction improves the privacy of planners.
- The average execution time is considerably reduced, by almost 50%;
- There is a substantial increase in the plan quality. This is probably due to the fact that, by holding states with outgoing novelty greater than 1, the priority is given to states that can be reached using a number of actions at most equal to the number of propositions of the problem, i.e., states that in the worst case can be reached by plans shorter than with outgoing novelty greater than 1.

Also, MA-BFWS with $w^{out} = 2$ leads to a sharp decrease in the average CPU time compared to MA-BFWS without novelty filtering. This is however less than with $w^{out} = 1$ for two reasons: (i) determining that the outgoing novelty for $w^{out} = 1$ is computationally much cheaper than for $w^{out} = 2$; (ii) MA-BFWS with $w^{out} = 1$ sends fewer states, and probably fewer superfluous states. Compared to MA-BFWS without novelty filtering, the average number of messages exchanged by MA-BFWS with $w^{out} = 2$ is lower, and the number of solved problem is higher (eight more than without filtering messages).

Figure 1 shows the percentage of sent messages w.r.t. the total amount of generated messages for the problems solved within different CPU time limits. We can observe that, compared to MA-BFWS without filtering of messages, for CPU time limits greater than 170 seconds, the use of outgoing novelty equal to 1 reduces by one order of magnitude the number of exchanged messages.

Domain	*secure*-MA-BFWS	*Secure* $w^{out} = 1$	*Secure* $w^{out} = 2$
From CoDMAP			
Blocksworld	**20**	**20**	**20**
Depot	**20**	17	**20**
DriverLog	**20**	**20**	**20**
Elevators	**20**	**20**	**20**
Logistics	**20**	**20**	**20**
Rovers	**20**	**20**	**20**
Satellites	**20**	**20**	**20**
Sokoban	11	13	**15**
Taxi	**20**	**20**	**20**
Wireless	**2**	**2**	**2**
Woodworking	11	**12**	**12**
Zenotravel	**20**	**20**	**20**
From MBS			
MA-BW	**19**	**19**	**19**
MA-BW-L	**15**	**15**	**15**
MA-Log	**20**	**20**	**20**
MA-Log-L	19	**20**	**20**

Performance	*secure*-MA-BFWS	*Secure* $w^{out} = 1$	*Secure* $w^{out} = 2$
Coverage (320)	277	278	**283**
Avg time	15.55	**5.79**	10.42
Avg quality	182.3	**172.56**	181.85
Avg k-messages	1036.47	**87.15**	525.73
Avg k-states	412.9	**184.23**	304.79
Time score	230.52	**257.88**	240.34
Quality score	233.76	**240.76**	239.73
Stdev Time	3.36	**1.37**	1.79
Stdev Quality	16.27	16.3	**16.06**

Table 5: Coverage (upper table) of *secure*-MABFWS without outgoing novelty filtering, with outgoing novelty w^{out} equal to 1 and 2 for domains of CoDMAP and MBS benchmarks; their performance (bottom table) in terms of number of solved problems, average CPU time (in seconds), plan cost, number of sent messages (in thousands), number of expanded states (in thousands), time score, quality score, standard deviation of the CPU times, and standard deviations of the plan costs. The best performances are indicated in bold.

Table 5 shows the performance of MA-BFWS with the "secure check" proposed by Brafman (2015) for the exchanged messages, denoted by *secure*-MABFWS. This version is incomplete but strong privacy-preserving over domains such as Logistics, Rovers and Satellites, where the heuristic function is independent from the private part of the problem (Brafman 2015), as it does not send states whose public projection has been sent before. Comparing the results with Table 4, we can observe a very limited decrease in terms of coverage and an improvement in terms of transmitted messages and expanded states.

We also compare our approach with other three existing approaches, MA-SIW, the approach mostly related to our work, PSM, and the best performing configuration of MAPLAN. The latter two planners were the best planners that took part in the CoDMAP competition. Table 6 shows the results of this comparison for the CoDMAP domains. As for benchmark MBS, MA-SIW solves no problem, while

Domain	MA-BFWS $w^{out}=1$	MA-SIW	MAPLAN	PSM
Blocksworld	**20**	**20**	**20**	**20**
Depot	17	8	12	17
DriverLog	**20**	**20**	16	**20**
Elevators	**20**	**20**	8	12
Logistics	**20**	18	18	18
Rovers	**20**	**20**	**20**	19
Satellites	**20**	**20**	**20**	13
Sokoban	13	4	**17**	16
Taxi	**20**	**20**	**20**	**20**
Wireless	2	0	**4**	0
Woodworking	12	1	15	**18**
Zenotravel	**20**	**20**	**20**	10
Overall (240)	204	171	190	184

Table 6: Number of problems solved by MA-BFWS with outgoing novelty filtering w^{out} equal to 1 w.r.t. MA-SIW, MAPLAN and PSM for the benchmark problems of CoDMAP. The best performances are indicated in bold.

MAPLAN and PSM do not support private goals, which are present in these problems. The time limit used for this comparison is 30 minutes, the same limit as in the competition. The results in Table 6 show that for the competition problems *secure*-MABFWS with $w^{out} = 1$ outperforms MA-SIW and is better than MAPLAN and PSM. Remarkably, the only type of information that agents share by using our approach is related to the exchanged search states, while MAPLAN also requires sharing the information for the computation of the search heuristics. In this sense, besides solving a larger set of problems, MA-BFWS exposes less private knowledge to other agents.

Messages delay

In this section, we present an experiment aimed at testing how MA-BFWS with filtering of messages performs in a heavily congested distributed network. With this aim, we introduced a mechanism that, by means of an artificial delay applied on each exchanged message, can simulate arbitrarily network delays during message transmission. These delays are distributed according to the gamma distribution that is an approximation of the delays in the Internet network (Sukhov and Kuznetsova 2009). In particular, Table 7 considers 5 different configurations of delays, with a standard deviation equal to 10% of the average delay.

For average delays smaller than 100 ms, in terms of coverage, we observe no significant performance gap w.r.t. MA-BFWS with no delay in the transmission of messages. Instead, as one could expect, with very high delays the number of solved problems decreases. However, even with an average delay of 10 seconds applied to each exchanged message, the number of solved problems does not decrease to few units. This probably indicates that many problems in our benchmarks does not require an intensive cooperation among agents.

Furthermore, we can observe a significant increase of the execution time when the delay increases. The required CPU time varies from 4.89 seconds for MA-BFWS without delay, to 162.92 seconds for the highest average delay we considered in our experiment. The higher number of expended states when the average delay is high is probably due to the fact that, in that case, the agents can spend more time for the state expansion. Moreover, concerning the plan quality, contrary to what one could expect, the best value is found with the greatest delay. A possible explanation of this behavior is related to the high number of expanded states, which helps to find better plans.

Domain	no delay	10ms	100ms	1s	10s
From CoDMAP					
Blocksworld	**20**	**20**	**20**	**20**	19
Depot	18	**19**	**19**	9	5
DriverLog	**20**	**20**	**20**	**20**	19
Elevators	**20**	**20**	**20**	**20**	19
Logistics	**20**	**20**	**20**	18	15
Rovers	**20**	**20**	**20**	**20**	**20**
Satellites	**20**	**20**	**20**	**20**	**20**
Sokoban	**14**	13	**14**	12	12
Taxi	**20**	**20**	18	0	0
Wireless	**2**	**2**	**2**	1	0
Woodworking	11	12	**13**	**13**	11
Zenotravel	**20**	**20**	**20**	19	**20**
From MBS					
MA-BW	**19**	**19**	**19**	**19**	**19**
MA-BW-L	**15**	**15**	**15**	**15**	**15**
MA-Log	**20**	**20**	**20**	**20**	0
MA-Log-L	**20**	**20**	**20**	**20**	0

Performance	no delay	10ms	100ms	1s	10s
Coverage (320)	279	**280**	**280**	246	194
Avg time	**4.89**	5.0	6.56	22.14	162.92
Avg quality	153.13	148.28	141.27	138.52	**136.26**
Avg k-messages	**17.94**	23.45	19.11	57.82	651.98
Avg k-states	**62.14**	71.59	92.76	365.39	1374.01
Time score	**265.14**	261.1	215.12	127.06	68.48
Quality score	234.17	241.08	**252.83**	218.65	167.95
Stdev Time	0.88	0.65	**0.62**	0.92	5.89
Stdev Quality	12.7	11.08	8.17	**7.18**	7.98

Table 7: Coverage (upper table) of MA-BFWS with no delay in the transmission of messages, with an average delay in the transmission of each message equal to 10 ms, 100ms, 1s, and 10s for domains of CoDMAP and MBS benchmarks; their performance (bottom table) in terms of number of solved problems, average CPU time (in seconds), plan cost, number of sent messages (in thousands), number of expanded states (in thousands), time score, quality score, standard deviation of the CPU times, and standard deviations of the plan costs. The best performances are indicated in bold.

Conclusion

In this paper, we show that novelty based techniques are very useful to significantly reduce the number of messages transmitted among agents, better preserving their privacy levels as well as improving their performance. The experimental analysis in this paper shows the effectiveness of our techniques in terms of coverage, CPU-time and number of transmitted messages. Moreover, we observed that our approach is robust to delays in the transmission of messages that could

occur in overloaded networks.

In future work we intend to explore the usage of privacy-preserving set operations for secure multi-party computation (Kissner and Song 2005), and the secure computation of joint novelty functions among agents. This may lead MA-BFWS to coordinate agents through a joint novelty table.

Acknowledgements

This research carried out with the support of resources of the National Collaborative Research Infrastructure Strategy (NeCTAR), and the Big & Open Data Innovation Laboratory (BODaI-Lab) of the University of Brescia, which is granted by Fondazione Cariplo and Regione Lombardia. Nir Lipovetzky, has been partially funded by DST group.

References

Bazzotti, G.; Gerevini, A. E.; Lipovetzky, N.; Percassi, F.; Saetti, A.; and Serina, I. 2018. Iterative width search for multi agent privacy-preserving planning. In *International Conference of the Italian Association for Artificial Intelligence*, 431–444. Springer.

Bonisoli, A.; Gerevini, A. E.; Saetti, A.; and Serina, I. 2018. A privacy-preserving model for multi-agent propositional planning. *Journal of Experimental & Theoretical Artificial Intelligence* 30(4):481–504.

Brafman, R. I., and Domshlak, C. 2008. From one to many: Planning for loosely coupled multi-agent systems. In *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling*, 28–35. ICAPS.

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

Gerevini, A. E.; Lipovetzky, N.; Percassi, F.; Saetti, A.; and Serina, I. 2019. Best-first width search for multi agent privacy-preserving planning. In *ICAPS*, to appear.

Kissner, L., and Song, D. 2005. Privacy-preserving set operations. In *Annual International Cryptology Conference*, 241–257. Springer.

Lipovetzky, N., and Geffner, H. 2012. Width and serialization of classical planning problems. In *ECAI 2012 - 20th European Conference on Artificial Intelligence*, 540–545.

Lipovetzky, N., and Geffner, H. 2017. Best-first width search: Exploration and exploitation in classical planning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 3590–3596. AAAI.

Maliah, S.; Brafman, R. I.; and Shani, G. 2017. Increased privacy with reduced communication in multi-agent planning. In *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling*, 209–217. ICAPS.

Maliah, S.; Shani, G.; and Stern, R. 2014. Privacy preserving landmark detection. In *Proceedings of European Conference on Artificial Intelligence*, volume 14. ECAI.

Maliah, S.; Shani, G.; and Stern, R. 2016. Stronger privacy preserving projections for multi-agent planning. In *Proceedings of the Fourth Workshop on Disrtibuted and Multi-Agent Planning*, 221–229. ICAPS.

Maliah, S.; Stern, R.; and Shani, G. 2016. Privacy preserving lama. In *Proceedings of the Fourth Workshop on Disrtibuted and Multi-Agent Planning*, 100–108. ICAPS.

Muise, C.; Lipovetzky, N.; and Ramirez, M. 2015. MAP-LAPKT: Omnipotent multi-agent planning via compilation to classical planning. In *Proc. of CoDMAP*.

Nissim, R., and Brafman, R. I. 2012. Multi-agent A* for parallel and distributed systems. In *Proceedings of the Workshop on Heuristics and Search for Domain-Independent Planning*, 42–51. ICAPS.

Nissim, R., and Brafman, R. I. 2014. Distributed heuristic forward search for multi-agent planning. *Journal of Artificial Intelligence Research* 51(1):293–332.

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

Sukhov, A. M., and Kuznetsova, N. Y. 2009. What type of distribution for packet delay in a global network should be used in the control theory? *CoRR* abs/0907.4468.

Sustrik, M. 2016. nanomsg. http://nanomsg.org/. [Online].

Torreño, A.; Onaindia, E.; and Sapena, Ó. 2014. Fmap: Distributed cooperative multi-agent planning. *Applied Intelligence* 41(2):606–626.

Tožička, J.; Štolba, M.; and Komenda, A. 2017. The limits of strong privacy preserving multi-agent planning. In *Twenty-Seventh International Conference on Automated Planning and Scheduling*.