# A Case For Domain-Independent Deterministic Multiagent Planning

Michal Štolba

stolba@agents.fel.cvut.cz

Department of Computer Science and Engineering,

Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Republic

***Abstract*—The notion of planning using multiple agents has been around since the very beginning of planning itself. It has been approached from various viewpoints especially in the multiagent systems community. Recently, domain-independent multiagent planning has gained more attention also in the automated planning community. In this paper, we shortly present the current state of the art, question some aspects of the research field and discuss the rising challenges.**

## I. Introduction

We could trace the first mention of *multiagent planning* back nearly as far as STRIPS itself – in 1980 Nillson published together with Konolige a paper titled "Multiple-Agent Planning Systems" [1], in which they presented a high-level extension of STRIPS [2] towards multiple agents. Since then, the topic has been active mainly in the multiagent systems community. One of the most cited works on distributed and multiagent planning is [3], which describes basics of possible coordination schemes for planning agents. There was also a large amount of work dealing with another facets of the coordination area, but usually without deeper study of the plan synthesis part (e.g., GPGP [4]) or requiring additional domain-specific knowledge (e.g., TALPlanner [5]). Multiagent planning also often relates to planning models described by Decentralized POMDPs (Dec-POMDPs) [6], similarly as POMDPs are used in single agent planning under uncertainty. The point of view on multiagent planning from the multiagent community is extensively summarized in [7], [8].

In the planning community, extensions of the PDDL language intended to support multiagent planning were introduced as Multiagent Planning Language (MAPL) in [9] and as Multiagent PDDL (MA-PDDL) in [10], but none of the extensions gained wide popularity, probably because of their complexity. In 2008, a well accepted paper on multiagent planning was published by Brafman and Domshlak at the ICAPS conference [11], which finally ignited more interest in the topic. The paper formally introduced a minimalistic extension of STRIPS (thus forming MA-STRIPS) and have shown that the complexity of multiagent planning is not directly exponentially dependent on the number of agents, but rather on the tree-width of their interaction graph and a minimal number of interactions needed to solve the problem. Such results suggested, that at least for loosely coupled problems (where the tree-width is low), the approach may be beneficial. In the following years, several multiagent planners were proposed [12], [13], [14], [15], [16], [17], [18], [19], either using directly the MA-STRIPS formalism, or some ad-hoc, but very similar one. Rather different approach was taken by Crosby et al. in [20], [21], where the agent decomposition was based not on actions, but on a variables in Finite Domain Representation, and was used in a centralized planner, significantly improving performance over (single agent) SOTA mainly in well decomposable domains. A dedicated workshop Distributed and Multiagent Planning (DMAP) also took place at the ICAPS'13 and ICAPS'14 conferences.

## II. MA-STRIPS

MA-STRIPS is the most commonly used formalism for domain-independent deterministic cooperative multiagent planning. The domain independence means, that the input of the planner consists of description of available operators, predicates, functions etc. describing the general mechanics of the world as well as the particular instance of the world represented by ground initial state and the particular problem represented by the ground goal facts that need to be achieved. Most commonly the input is described using PDDL language, where the general part is termed "domain" and is described in a domain file and the particular initial and goal configurations are termed "problem" and are described in a problem file. In the multiagent setting, additional information is typically needed to determine what are agents and which actions belong to which agent (this is not part of the MA-STRIPS formalism).

Deterministic in this case relates to the action model, where each action has deterministic effects, although the multiagent planning algorithm itself may be non-deterministic in the sense that for the same input it gives different output and runs for different time - this is caused by the inherent nondeterminism of communication and distributed computation.

By cooperative multiagent planning we understand a setting where a group of agents is attempting to find a joint plan, where each agent uses its own actions in order to reach some joint (and possibly some private) goals. The agents typically need to coordinate their actions and to cooperate in order to do so, although each agent is planning for itself and the agents may have some private knowledge which it does not want to reveal to other agents.

The MA-STRIPS formalism is very much the same as the original STRIPS, except, that the actions are disjunctively split (or factored) among the agents, resulting in the following definition of the multiagent planning problem:

**Definition 1. Multiagent planning problem** is a quadruple $\Pi = \langle \mathcal{L}, \mathcal{A}, s_0, S_g \rangle$, where $\mathcal{L}$ is a set of propositions, $\mathcal{A}$ is a set of *agents* $\alpha_1, \ldots, \alpha_{|\mathcal{A}|}$, $s_0$ is an initial state and $S_g$ is a set of goal states. An *agent* $\alpha = \{a_1, \ldots, a_n\}$ is represented by a set of actions it can perform. A *state* $s \subseteq \mathcal{L}$ is a set of atoms from a finite set of propositions $\mathcal{L} = \{p_1, \ldots, p_m\}$ which hold in $s$. An *action* is a tuple $a = \langle \mathsf{pre}(a), \mathsf{add}(a), \mathsf{del}(a) \rangle$, where $a$ is a unique action label and $\mathsf{pre}(a), \mathsf{add}(a), \mathsf{del}(a)$ respectively denote the sets of preconditions, add effects and delete effects of $a$ from $\mathcal{L}$.

In the set of propositions $\mathcal{L}$ can further be identified subsets of *public* propositions known to all agents and $\alpha$-*internal* (a.k.a. $\alpha$-*private*) propositions known only to a specific agent $\alpha$. Such separation of the propositions can be derived from the problem itself (as proposed in [11]) , where a proposition is public iff it is used by any two actions of different agents and it is *internal* to the agent $\alpha$ ($\alpha$-*internal*) iff it is used only in actions of agent $\alpha$. Alternatively, what propositions are public or private can be a part of the problem definition (used in [16], [17]). How the actions are partitioned among agents is not a part of the MA-STRIPS definition and is implementation dependent, in most planners, the agents are defined as objects in PDDL and an action is assigned to an agent if the agent object is part of the grounded parameters of the action.

Similarly to the separation of the propositions, the actions of a single agent can be seen as either *public* or *internal*. An action $a \in \alpha$ is *internal* iff it uses only $\alpha$-*internal* propositions in its preconditions and effects. Otherwise, the action is *public*, which in fact means, that the action interacts with some other agent, or agents.

Following the STRIPS formalism, a solution for the multi-agent planning problem is a set of sequential plans (sequence of actions, one plan for each agent), where some of the actions of different agents may be performed in parallel, although in most implementations the resulting plan is sequential (i.e. there are no parallel actions). Formally, the solution is defined as follows:

**Definition 2.** A solution to Multiagent planning problem $\prod$ is a **multiagent plan** $P = \left\langle P_{\alpha_1}, ..., P_{\alpha_{|\mathcal{A}|}} \right\rangle$, where each $P_{\alpha_i}$ is a plan of agent $\alpha_i$, i.e. consisting of a sequence of actions in $\alpha_i \cup \{noop\}$. A multiagent plan P is valid, iff

1) For each $\alpha, \beta \in \mathcal{A}$, $|P_\alpha| = |P_\beta|$. The $noop$ actions mean that the agent is idle in given time-step.
2) For each time-step $t_k$ where $k \in \langle 1, ..., |P_\alpha| \rangle$, actions $a_i = P_{\alpha_i}[t_k]$ for all $i \in \langle 1, ..., |\mathcal{A}| \rangle$are not mutually exclusive.
3) All actions in time-step $t_1$ are applicable in the initial state, for all time-steps $t_k$ where $k \in \langle 2, ..., |P_\alpha| \rangle$, actions in $t_k$ are applicable in the state resulting from time-step $t_{k-1}$ and the state resulting from time-step $t_{|P_\alpha|}$ is in the set of goal states $S_g$.

Since all the plans are required to have the same length, the MA plan can be represented by a matrix, where each row consists of a plan for single agent and each column contains all actions in the respective time-step.

It is clear that the MA-STRIPS formalism is very simplistic and minimalistic extension and can be developed further to accommodate various aspects of the multiagent systems (i.e., finer separation of *public* and *internal* actions, as discussed for example in [22]). The philosophy of the formalism is to provide the smallest possible common ground and to enable the easiest possible migration of the planning algorithm to the multiagent setting.

Unlike in many multiagent systems, the MA-STRIPS formalism assumes the agents to be fully cooperative. From the multiagent perspective, this may seem as an oversimplification, but similarly to the simplicity of the MA-STRIPS, we argue to first tackle the problems of the simplest possible multiagent extension – cooperative agents – and then gradually add complexities of the multiagent nature, such as selfish agents, negotiation and such. Some attempts have already been taken, for example in [23], [24].

## III. Is it any Good?

The first question that is asked after introduction of some new or nonstandard paradigm is – what is it good for? How does it help us? In this paper, we would like to ask such questions and try to come up with some answers. From the multiagent point of view, we can see significant benefits of introducing domain-independent planning into multiagent systems, namely reuse of the wide spectrum of planning systems and techniques from the area of classical planning. On the other hand, the drawback of MA-STRIPS for the multiagent setting is that it requires cooperation among the agents. This can possibly be mitigated in future by means such as mechanism design, as shown in [24].

In the following section, we will present some of the benefits we suggest that multiagent planning may bring to the area of planning in general.

### A. Improve scalability by decomposition

Although being largely improved in recent years, scalability is still an issue for domain-independent planning. Multiagent planning can be seen as a method of factorization of the planning problems, which has been studied in works such as [25], [26]. It has been shown that factorization of the planning problem can be beneficial, especially in the problems, where it is possible to find large independent parts – loosely coupled problems. Take for example the rovers domain with several rovers. The problem may be factored in such a way that the rovers do not need to coordinate, except for the last bit, where they use shared communication channel to communicate the result. In such a case, most of the planning can be done independently (even on different machines), however some coordination is needed here (it is not possible to simply run several classical planners in parallel). This is a case for multiagent planning.

Given a single agent problem (e.g. from the IPC benchmark set), it is not obvious what entities should be considered as

agents and how the factorization should be done. In MA-STRIPS, the first question is not answered, but the second one has some theoretical background. Once the actions are assigned to agents, the factorization is based on the interactions among the agents' actions, effectively meaning that it is based on the *causal graphs* of the problem (as in [11]), the factoring is one of the best known to date, as shown in [25]. Still, how to best assign actions to agents is an open question.

In the FMAP planner [17], [27] formalism, the actions are assigned to agents by the domain designer, as is the separation of private and shared information. This gives the domain designer more freedom, but also more responsibility in that making more information public may increase the problem complexity (as the tree width of interaction graph rises) and making less information public may render the problem unsolvable (i.e. global solution may not exist).

Completely different approach to factorization was taken by Crosby in [20], [21], where the factorization is automatic and based on the variables of a Finite Domain Representation (FDR) of the planning problem. The method is, again, based on Causal Graph structures and tries to identify agents as sets of variables which are somehow self-contained and represent the agent's internal state. Variables which do not form an internal state of any agent are understood as environment and thus public variables. Classification of actions is rather more complex than in MA-STRIPS, but can be summarized as follows. Actions, which interact only with the agent variables are assigned to the particular agent, other actions are left as public, which means that they can be used by any agent. This approach was not shown to increase scalability, but was shown to increase solution efficiency, especially in some domains. Note, that Crosby's approach was implemented as a centralized, single-thread planner.

Various kinds of factoring, and understanding gained from the research of it, may be also used for other aspects of classical planning, such as search state pruning [28].

### B. Multi-core / Cluster computing

Planning on multi-core machines gained wide popularity and had a separate track at the IPC2011 competition. It is definitely beneficial to be able to scale the computation using multiple cores. But, if we want to scale the computation even further and use multiple *machines*, we are no longer able to use shared memory (as in parallel computation). In such case, approaches of distributed computing utilizing message passing need to take place, and multiagent planning aspires to be one of them (although obviously not the only one). In this scenario, multiagent planning represented by MA-STRIPS can be seen as an equivalent for distributed planning with factorization fixed by the partitioning of actions to particular agents. Such approach has been successfully tested against SOTA in multi-core planning in [14] as a parallel version of MAD-A*, called MAP-A*. Again, the question of best factorization arises here, with the Crosby's variable based factorization being another candidate, however never tested in a distributed system.

### C. Privacy

In multiagent systems, privacy of the data is often one of the main concerns. In MA-STRIPS-based multiagent planning, this requirement is not one of the top priorities – the privacy (or *internality*) of the propositions and actions is understood rather as a way of improving effectiveness of the computation by reducing the complexity – but nonetheless if required by the application, some degree of privacy of the data can be achieved. In practice, it may be somewhat harder not to reveal the *structure* of the private information and it may also reduce usability of some techniques, such as distributed heuristics, but it seems to be possible. One such approach was described in [22], but without implementation or experimental evaluation.

### D. Asynchronous computation

Distributed computation (and multiagent as well) is inherently asynchronous and non-deterministic. This brings several challenges and complexities that need to be tackled, but once solved, some of the insights may be beneficial to classical planning as well. As a representative example, we can take the computation of a global heuristic.

A heuristic in multiagent planning can be computed locally, using only a projection of the planning problem to the propositions of the respective agent. A public action $a \in \alpha$ can be projected to agent $\beta$ by removing all $\alpha$-*internal* propositions from $\mathsf{pre}(a), \mathsf{add}(a)$ and $\mathsf{del}(a)$, thus retaining only *public* propositions (and technically also $\beta$-*internal* propositions, but actions of agent $\alpha$ contain none). Such heuristics were used for example in [14]. A different approach is to attempt to compute a global heuristic estimate *without* exposing the internal information to other agents. This can be achieved for example by requesting other agents for their estimates of the current state and the goal, or some sub-goals, as in [29], [30].

Such global heuristic computation may involve nontrivial communication, resulting in an asynchronous computation of the heuristic estimate. To our knowledge, such phenomenon has not been studied yet. It seems to be interesting also from the point of view of classical planning, because if successfully utilized, asynchronous computation of heuristics may be used to evaluate computationally intensive heuristics, to use external solvers for heuristic estimation, sensory input (in robotics), etc. Also it is interesting to observe, that the global heuristic is always equally or more informative than the projected heuristic, thus we have two heuristics, the first one is less informative, gives lower estimates, but is faster, the second one is more informative, gives higher estimates, but takes significantly longer to compute. If generalized, a sequence of gradually more precise, but costlier heuristics could be used to guide search (including blind search as the fastest one), which is an interesting research topic. Also, as the costly distributed heuristic is computed partly by other agents, the time while waiting for the results may be used to do some more computation such as exploration using only the local heuristic.

### E. Applications

MA-STRIPS planning may bring domain independent planning to some (more or less) new application domains. As such we see the following:

- **Multi-robotics:** Slowly but steadily, domain-independent planning is penetrating the robotic research, providing the robots with high-level reasoning. On the other hand, robotic research is being extended towards multi-robotic teams. Multiagent planning seems to bridge those two advancements and in the future may enable multi-robotic teams to be controlled by a high-level distributed planning system. In such scenarios, each robot would have its own planning representation of the world, thus reducing the domain and problem size and keeping the local information (i.e., sensory data) local. Communication will clearly be an issue, which may encourage research of multiagent planning techniques aiming not primarily on planning speed, but on the communication requirements (this may for example discourage the use of global heuristic estimates).
- **Orchestration of Internet services:** Since Internet services are computer programs already described in (formal) programming languages their coordination and orchestration can straightforwardly profit from multiagent planning techniques. Moreover services are usually described by interfaces resembling planning actions with preconditions on the input data and effect on the data. Therefore if such actions have to be ordered in a suitable fashion, beginning with initial constraints on the data and with a goal form of the data, planning is an appropriate fit. Additionally, the services can be distributed over the Internet and the data processing tasks can vary in the sense of particular areas of focus (e.g., financial processing, logistics services, etc.). Both of these challenges are covered by the domain-independent multiagent planning. A single agent planning applications of this kind were already covered for example in [31].
- **Coalition planning:** In recent years, many military operations involve coalitions. Coalition operations are in nature cooperative, but nevertheless, some information cannot be disclosed among the coalition partners. This can be generally modeled by multiagent systems and in particular by multiagent planning. Such a scenario is not applicable only in military, but may find uses also in business cooperation and other areas, although some more detailed privaci concerns may be necessary as illustrated in [22].

## IV. Challenges

In the following, we will present some of the outstanding challenges of domain-independent multiagent planning we are aware of.

### A. Comparison of Planners

Up to the present day, we know of about a half dozen domain-independent multiagent planners [12], [13], [14], [15], [16], [17] and more are in the development. The biggest issue in order to compare them is that although the underlying formalisms are mostly similar, the actual language used as input differs (it is PDDL with some ad-hoc definition of agents and/or public propositions). Some consensus on this matter needs to be settled.

Similarly to the common definition language, a widely accepted set of benchmarks is needed. In the recent works, the benchmarks were typically created by ad-hoc converting some suitable IPC domains, where the agent decomposition is natural, such as logistics or rovers. Only few new – multiagent specific – domains were introduced, i.e., cooperative pathfinding. In the future, it would be interesting to come up with more such domains, drawing from real world and multiagent systems applications and also designed to test some specific properties and pitfalls of multiagent planning. Naturally, the best venue for such efforts would be a dedicated IPC track.

The metrics used for planner comparison are also challenging. All metrics used to compare classical planners still apply, but there are some additional ones, such as number of exchanged messages or the amount of communicated data. Further metrics can be adopted from the research of other multiagent algorithms.

Moreover, the comparison of multiagent planners gains a new complexity in that a single problem can be partitioned in multiple ways, and each partitioning may be beneficial for different planners. Similarly, the amount of agent interaction significantly influences performance of the planners - some algorithms may be better suited for loosely coupled problems while other for tightly coupled problems. This should be also reflected in the potential analysis.

### B. Which Planning Paradigm is the Best?

In classical sequential domain-independent planning, the dominating planning paradigm seems to be heuristic forward-chaining search typically utilizing one or more highly informative heuristic estimators (used by planners as FastForward, FastDownward, LAMA and many more), although some other approaches (such as bidirectional search) has performed well in the last IPC competition. It is not clear, whether the same holds for multiagent planning (and for all metrics). Again, the only way to reliably answer this question is to perform a rigorous comparison of the planners, such as in the IPC competition.

Currently the most frequent approaches to multiagent planning are distributed heuristic search [14], [29], [30], partial order planning [16], [17], and various other, such as plan reuse [12] or generate-and-test [18], [19].

### C. Heuristics

One interesting area of research are the heuristic estimator for multiagent planning. As mentioned in section about asynchronous computation, the baseline solution for heuristic computation is to use a projection of the problem to the particular agent's propositions. This approach was used in [13], [14]. Another approach is to attempt to compute (or at least approximate) the global value as if the problem was not

partitioned in order to obtain better heuristic guidance. To our knowledge, the only heuristics treated this way so far were the FF heuristic in [29] and [30] and a Domain Transition Graph based heuristic in [17]. Of course, in some cases, the communication needed to compute a global heuristic may be prohibitive, in such situation, the local heuristic would have to suffice, or a clever combination of both would need to be devised.

Different challenge is to come up with a heuristic specific to the multiagent planning. Such heuristic may not only lead the search towards the goal, but may also lead the search in a way minimizing some multiagent metric, such as the communication load.

### D. Privacy

Multiagent planning can be seen also from the perspective of privacy-preserving distributed computing, although it is not the main concern of most of the MA-STRIPS based planners. This perspective raises many questions, some of them were already addressed in [22], [32], but a comprehensive analysis of possible privacy violations in MA-STRIPS and MA-STRIPS based planning algorithms has not been yet presented. Even the question if a MA-STRIPS based multiagent planner can be privacy preserving (and if so how strongly) is yet to be answered.

### E. Extending Complexity Analysis

From the theoretical perspective, the original [11]'s complexity results can give a solid groundwork for future and more detailed studies of complexity of multiagent planning. We envision two main directions: (i) problem-specific and (ii) finer complexity classes. The problem-specific study could be inspired by works as [33], where particular planning domains are analyzed from perspective of computational complexity showing which problems are easy, even if planning in general is known to be computationally hard. The idea of study of finer complexity classes, is for example parametrized analysis of classical planning by [34]. Additionally, these studies can be extended by communication complexity showing how much communication would be required to solve the problems.

## V. Final Remarks

Domain independent planning is a field of research bridging two distinct topics of multiagent systems and domain independent planning. We see providing a common problem definition language and set of benchmarks in order to be able to determine the state of the art and compare it with newly emerging techniques as one of the important steps to advance the field. We see such hybrid field to be potentially beneficial to both research communities.

*Acknowledgments:* This research was supported by the Czech Science Foundation (13-22125S).

## References

[1] K. Konolige and N. J. Nilsson, "Multiple-agent planning systems," in *AAAI*, vol. 80, 1980, pp. 138–142.

[2] R. Fikes and N. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," in *Proc. of the 2nd International Joint Conference on Artificial Intelligence*, 1971, pp. 608–620.

[3] E. H. Durfee, "Distributed problem solving and planning," in *A Modern Approach to Distributed Artificial Intelligence*, G. Weiß, Ed. San Francisco, CA: The MIT Press, 1999, ch. 3.

[4] K. Decker and V. Lesser, "Generalizing the partial global planning algorithm," *International Journal on Intelligent Cooperative Information Systems*, vol. 1, no. 2, pp. 319–346, June 1992. [Online]. Available: http://mas.cs.umass.edu/paper/34

[5] P. Doherty and J. Kvarnström, "TALplanner: A temporal logic-based planner," *AI Magazine*, vol. 22, no. 3, pp. 95–102, 2001.

[6] R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman, "Solving transition independent decentralized Markov decision processes," *Journal of Artificial Intelligence Research*, vol. 22, pp. 423–455, 2004. [Online]. Available: http://rbr.cs.umass.edu/shlomo/papers/BZLGjair04.html

[7] M. De Weerdt, A. Ter Mors, and C. Witteveen, "Multi-agent planning: An introduction to planning and coordination," in *Handouts of the European Agent Summer*, 2005.

[8] M. de Weerdt and B. Clement, "Introduction to planning in multiagent systems," *Multiagent and Grid Systems*, vol. 5, no. 4, pp. 345–355, 2009.

[9] M. Brenner, "A multiagent planning language," *Workshop on PDDL, ICAPS'03, Trento, Italy, 2003.*, p. 33, 2003.

[10] D. L. Kovacs, "A multi-agent extension of PDDL3.1," in *Proc. of the 3rd Workshop on the International Planning Competition (IPC)*, 2012, pp. 19–27.

[11] R. I. Brafman and C. Domshlak, "From one to many: Planning for loosely coupled multi-agent systems." in *ICAPS*, 2008, pp. 28–35.

[12] D. Borrajo, "Plan sharing for multi-agent planning," in *Proc. of DMAP Workshop of ICAPS'13*, 2013, pp. 57–65.

[13] R. Nissim, R. I. Brafman, and C. Domshlak, "A general, fully distributed multi-agent planning algorithm," in *Proc. of AAMAS'10*, 2010, pp. 1323–1330.

[14] R. Nissim and R. I. Brafman, "Multi-agent A* for parallel and distributed systems," in *Proc. of AAMAS'12*, ser. AAMAS '12, Richland, SC, 2012, pp. 1265–1266. [Online]. Available: http://dl.acm.org/citation.cfm?id=2343896.2343955

[15] D. Pellier, "Distributed planning through graph merging." in *ICAART (2)*, 2010, pp. 128–134.

[16] A. Torreño, E. Onaindia, and O. Sapena, "An approach to multi-agent planning with incomplete information," in *Proc. of ECAI*, 2012, pp. 762–767.

[17] A. Torreño, E. Onaindia, and O. Sapena, "Fmap: a heuristic approach to cooperative multi-agent planning," in *Proc. of DMAP Workshop of ICAPS'13*, 2013, pp. 84–92.

[18] J. Tožička, J. Jakubuv, K. Durkota, and A. Komenda, "Multiagent planning by iterative negotiation over distributed planning graphs," in *Proc. of DMAP Workshop of ICAPS'14*, 2014, pp. 7–15.

[19] J. Tožička, J. Jakubuv, K. Durkota, A. Komenda, and M. Pěchouček, "Multiagent planning supported by plan diversity metrics and landmark actions," in *Proc. of ICAART'14*, 2014.

[20] M. Crosby, M. Rovatsos, and R. Petrick, "Automated agent decomposition for classical planning," in *Proc. of ICAPS'13*, 2013. [Online]. Available: https://www.aaai.org/ocs/index.php/ICAPS/ICAPS13/paper/view/6051

[21] M. Crosby, "Multiagent classical planning," Ph.D. dissertation, PhD Thesis, University of Edinburgh, 2014.

[22] A. Bonisoli, A. Gerevini, A. Saetti, and I. Serina, "A privacy-preserving model for the multi-agent propositional planning problem," in *Proc. of DMAP Workshop of ICAPS'14*, 2014, pp. 25–29.

[23] M. Crosby and M. Rovatsos, "Heuristic multiagent planning with self-interested agents," in *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*. International Foundation for Autonomous Agents and Multiagent Systems, 2011, pp. 1213–1214.

[24] R. Nissim and R. I. Brafman, "Cost-optimal planning by self-interested agents," in *Proc. of DMAP Workshop of ICAPS'13*, 2013, pp. 1–7.

[25] R. I. Brafman, "Factored planning: How, when, and when not," in *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-2006*, 2006, pp. 809–814.

[26] E. Fabre, L. Jezequel, P. Haslum, and S. Thiébaux, "Cost-optimal factored planning: Promises and pitfalls." in *ICAPS*, 2010, pp. 65–72.

[27] A. Torreno, E. Onaindia, and O. Sapena, "FMAP: Distributed cooperative multi-agent planning," *Applied Intelligence*, 2014.

[28] R. Nissim, U. Apsel, and R. I. Brafman, "Tunneling and decomposition-based state reduction for optimal planning." in *ECAI*, 2012, pp. 624–629.

[29] M. Štolba and A. Komenda, "Fast-forward heuristic for multiagent planning," in *Proc. of DMAP Workshop of ICAPS'13*, 2013, pp. 75–83.

[30] M. Stolba and A. Komenda, "Relaxation heuristics for multiagent planning," in *Proc. of ICAPS'14*, 2014, pp. 7–15.

[31] B. Srivastava and J. Koehler, "Web service composition - current solutions and open problems," in *ICAPS 2003 Workshop on Planning for Web Services*, 2003, pp. 28–35.

[32] R. Nissim and R. Brafman, "Distributed heuristic forward search for multi-agent systems," *CoRR/arXiv*, 2013.

[33] M. Helmert, "New complexity results for classical planning benchmarks," in *ICAPS*, D. Long, S. F. Smith, D. Borrajo, and L. McCluskey, Eds. AAAI, 2006, pp. 52–62.

[34] C. Bäckström, Y. Chen, P. Jonsson, S. Ordyniak, and S. Szeider, "The complexity of planning revisited - a parameterized analysis," in *AAAI*, J. Hoffmann and B. Selman, Eds. AAAI Press, 2012.