MAPlan

Daniel Fišer and **Michal Štolba** and **Antonín Komenda**
Department of Computer Science, Faculty of Electrical Engineering,
Czech Technical University in Prague, Czech Republic
danfis@danfis.cz, {*stolba,komenda*}*@agents.fel.cvut.cz*

Abstract

In this paper, we present a brand new multi-agent planner called MAPlan. The planner implements state space heuristic search on factorized problems retrieved from either unfactored or factored version of MA-PDDL files. It can run both in a multi-thread and distributed configuration with a communication over network or within a process. This paper briefly describes the details of the MAPlan configurations used in the 2015 CoDMAP competition.

Introduction

MAPlan is a multi-agent planner implementing multi-threaded as well as distributed state space heuristic search. The MAPlan planner further expands on the ideas introduced in the MAD-A* planner (Nissim and Brafman 2012). The basic search scheme is following. All operators retrieved from the SAS$^+$ representation of the problem are divided between individual agents so that each operator belongs to only one agent. Each agent expands the state space only by its own operators, but whenever a public operator is used, the resulting state (public state) is sent to all other agents. As the public operators are considered those that have some common facts in their preconditions or effects with operators owned by other agents. So the planner can explore the state space with its own operators without communication with any other agent if the results of the applied operators cannot influence others, but the states useful to other agents are shared among all agents.

Moreover, an agent can have access to the projected operators if it is needed for a computation of heuristics. The projected operator is an image of the public operator owned by the other agent that preserves only those preconditions and effects that are already known to the agent. In other words, each agent is aware only of those facts that are in the preconditions and effects of its own operators and the projected operators are images of other agents' public operators with deleted facts that the agent cannot understand.

The planner is implemented in C and is able to run a multi-threaded as well as distributed search. The communication between agents can be inner process in case of multi-threaded search or over TCP/IP protocol. Both factored and unfactored versions of MA-PDDL files can be used as input problem definitions. In the following three sections, the heuristics used in the configurations submitted to the 2015 CoDMAP competition are described and more details about the planner specific to the centralized and the distributed track are provided.

Heuristics

MAPlan planner uses altogether four different heuristics in configurations submitted to 2015 CoDMAP competition. The first one is the LM-cut heuristic (Bonet and Helmert 2010). The heuristic runs on projected operators that are extracted either from unfactored or factored version of MA-PDDL domain definitions.

The second one is the distributed version of LM-cut heuristic (Štolba, Fišer, and Komenda 2015a) which was designed to provide provably equal estimates as the centralized version of LM-cut heuristic on all operators not only the projected ones. This property comes with a cost of increased computational burden causing less expanded states per time unit. Nevertheless, the heuristic provides more accurate estimates than LM-cut on projected operators which should compensate the disadvantage of a slower state expansion. Both versions of LM-cut heuristics are admissible so they are used in the optimal planning track.

The next one is the distributed Fast-Forward (FF) heuristic (Štolba and Komenda 2014), specifically the lazy variant. The computation of a heuristic estimate starts with an exploration of the relaxed problem on projected operators. The following step is an extraction of the relaxed plan which is performed in a distributed manner. Once a projected operator is reached during the extraction, the owner of the operator is asked to provide its local part of the relaxed plan leading to that operator. If the local part of the plan contains a projected operator, the path to that operator must be extracted too. This would lead to a distributed recursion. The implementation used in MAPlan avoids a distributed recursion by collecting the local parts of the relaxed plan by the initial agent and requesting the other agents consecutively until all projected operators are not solved.

The last heuristic function used in CoDMAP competition is the distributed version of FF heuristic based on Domain Transition Graphs (DTGs) (Štolba, Fišer, and Komenda 2015b). This heuristic is based on an exploration of agents'

local DTGs constructed from projected operators. The unknown (or more precisely possibly unknown) preconditions and effects of projected operators are recorded into DTGs with a special symbol. Once that symbol is reached during extraction of the relaxed plan, a distributed recursion is executed and the cost of the relaxed plan is computed with the help of the owners of the projected operators. Moreover, partial plans for each fact can be cached and later reused without re-computation of the full estimate. It makes this heuristic very fast even though it is built upon a distributed recursion.

Centralized Track

MAPlan configuration for the centralized track uses the unfactored version of MA-PDDL definition files. A factorization of a problem to particular agents is made as suggested in the MA-PDDL files. The definition files are translated to SAS^+ representation by a slightly modified translator from fast-downward planner (Helmert 2006). The factorization itself is made by MAPlan from SAS^+ representation before any agent is started and it is based on splitting the operators according to an emergence of particular agents' names in parameters of operators. Privateness of facts and operators is inferred also from SAS^+ representation. The facts that are stated only in preconditions or effects of operators of only a single agent are considered as private and the operators that have only private preconditions and effects are considered as private. Similarly, projected operators are created simply by omitting the private facts from preconditions and effects.

Since the translation from MA-PDDL is performed in a centralized manner before any agent is started, all agents share the same representation of a state. This considerably simplifies a communication of public states between agents because each agent can directly use the received state without any additional processing. The disadvantage of this approach, of course, resides in insufficient preservation of private information because private parts of public states are freely communicated between agents. This could be easily solved if the private parts would be somehow scrambled when transmitted and the receiving agent would be able to unscramble its own private parts. Nevertheless, even though this (or any similar) mechanism is not implemented in the planner, each agent "understands" only its own private facts and does not use any private information of other agents in any way. So the privateness is at least preserved in this way. In fact, we consider this problem only as an implementation detail, but of course this property should be considered in a comparison with other planners.

In centralized track, each agent runs in a separate thread and communication channels between agents are created within a common process context. Two configurations for optimal planning are submitted, both using A^* search algorithm – one with the projected LM-cut heuristic and one with the distributed LM-cut heuristic. One configuration for satisficing planning with best-first search algorithm is submitted where both inadmissible heuristics, distributed FF and distributed DTG-based FF, are used, each for a half of the maximum allowed time. All planners are complete.

Distributed Track

For the distributed track, the factored version of MA-PDDL files is used. In contrast to the centralized track, the translation to SAS^+ has to be done separately by each agent from its own MA-PDDL factor. As in the previous case, the translation is done by the translate tool from fast-downward planner but this time it had to be modified more than slightly. The translation to SAS^+ has to be distributed over all agents because particular factors are available only to the corresponding agents and it is not possible for a single agent to perform concise grounding of the problem only from its own factor.

The translation consists of two main phases. In the first phase, the concise grounding of the problem is made by a coordinate effort of all agents that communicate in a ring, i.e., an absolute ordering of agents must be provided and each agent sends messages only to the agent that is next in the ordering (and the last agent sends messages to the first one in ring). This way, the messages circle around the established ring of agents. The grounding starts with the first agent in the ring, which uses a Datalog program (Helmert 2009) implemented in fast-downward's translate tool for grounding of its local problem. All public facts that are returned by the Datalog program are sent to the next agent. The next agent adds the received public facts to the initial state and continues with the same procedure. It runs the Datalog program and the public facts from its output sends to the next agent. This whole procedure continues until the first agent in the ring does not receive the same public facts that it already transmitted. In this moment all agents know all grounded facts that are public.

In the second phase, SAS^+ variables and their values must be inferred from the public and also private facts. This is done from fact invariants (Helmert 2009). This might be a little bit tricky in this case because each agent can identify different invariants that can even overlap each other. So the invariants viable for all agents are identified via a distributed coordination of agents. The ring of agents that is already established is used and the first agent sends its invariants of the public facts to the next agent. The next agent uses its own invariants to split the received invariants to preserve an invariant property and so on. At the end of this procedure all agents have the same invariants of public facts.

The resulting SAS^+ variables are created so that all agents share the exactly same representation of the public part of a state and the private parts differ. In other words, the private facts translate to a separate variables. Although some private facts could share a variable with some public facts because they can be an invariant together, this design considerably simplifies communication of states between agents. This way, all agents share the exactly same representation of the public part of a state but each agent still can privately manage the private variables without communicating it directly to other agents.

The obvious disadvantage of this approach is that each agent must somehow reconstruct the full state with its own private part from the received public state. In MAPlan, this problem is solved by attaching an identification of the full state to the public part that is sent to other agents. The receiving agent must preserve this identification and send it

along the next state that is created via expansions from the original received state. Thus sets of state identifications from all agents travel as tokens with all public states that are communicated and the receiving agent can always reconstruct the full state, i.e., find out its private part for the state. The same approach is used for distributed heuristics whenever a state has to be sent to other agent.

The advantage is that the privacy of states is preserved because private information is never transmitted; not even during translation of the problem to SAS$^+$. The only transmitted information linked to the private part of a state is its identification number. Nevertheless, that number has no meaning to any other agent than the one that created it.

The agents in the distributed track run in separate processes and communicate over network via TCP/IP. The configurations are the same as in the case of the centralized track.

Conclusion

In this paper, a brand new multi-agent planner, called MA-Plan, was introduced. The planner is based on ideas introduced by the MAD-A* planner and it is implemented in C. The planner accepts both factored and unfactored versions of MA-PDDL and it can utilize two different privacy preserving schemes with their different advantages and disadvantages. The planner is complete and it can be used both as an optimal planner and a satisficing planner. It can run in one process as a multi-threaded application or it can be distributed over a network of computers. Although the planner implements a moderate set of heuristics that can be used locally with projected operators, it also contains a rich set of distributed heuristics.

Acknowledgments

This research was supported by the Czech Science Foundation (grant no. 15-20433Y), the Grant Agency of the CTU in Prague (grant no. SGS14/202/OHK3/3T/13) and USAF EOARD (FA8655-12-1-2096). Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the program Projects of Large Infrastructure for Research, Development, and Innovations (LM2010005), is greatly appreciated.

References

Bonet, B., and Helmert, M. 2010. Strengthening landmark heuristics via hitting sets. In *ECAI*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, 329–334. IOS Press.

Helmert, M. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.

Helmert, M. 2009. Concise finite-domain representations for pddl planning tasks. *Artificial Intelligence* 173(5-6):503–535.

Nissim, R., and Brafman, R. I. 2012. Multi-agent a* for parallel and distributed systems. In *Proc. of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'12)*, 1265–1266.

Štolba, M., and Komenda, A. 2014. Relaxation heuristics for multiagent planning. In *Proc. of the 24th International Conference on Automated Planning and Scheduling (ICAPS'14)*, 298–306.

Štolba, M.; Fišer, D.; and Komenda, A. 2015a. Admissible landmark heuristic for multi-agent planning. In *Proc. of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15)*.

Štolba, M.; Fišer, D.; and Komenda, A. 2015b. Comparison of RPG-based FF and DTG-based FF disrtibuted heuristics. In *Proc. of the 3rd Workshop on Distributed and Multi-Agent Planning (DMAP)*.