Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling (ICAPS 2017)

Initial Results on Generating Macro Actions from a Plan Database for Planning on Autonomous Mobile Robots

Till Hofmann, Tim Niemueller, Gerhard Lakemeyer
Knowledge-Based Systems Group
RWTH Aachen University, Germany
{hofmann, niemueller, lakemeyer}@kbsg.rwth-aachen.de

Abstract

Planning in an on-line robotics context has the specific requirement of a short planning duration. A property of typical contemporary scenarios is that (mobile) robots perform similar or even repeating tasks during operation. With these robot domains in mind, we propose database-driven macro planning for STRIPS (DBMP/S) that learns macros – action sequences that frequently appear in plans – from experience for PDDL-based planners. Planning duration is improved over time by off-line processing of seed plans using a scalable database. The approach is indifferent about the specific planner by representing the resulting macros again as actions with preconditions and effects determined based on the actions contained in the macro. For some domains we have used separate planners for learning and execution exploiting their respective strengths. Initial results based on some IPC domains and a logistic robot scenario show significantly improved (over non-macro planners) or slightly better and comparable (to existing macro planners) performance.

1 Introduction

Task planning is not as widespread in robotics as could be assumed by the expected benefits of declarative specifications, automated action sequencing, and extensibility. Long planning times (at run-time) are a major issue for many real-time and on-line scenarios. Especially continual planning (Brenner and Nebel 2009; Hofmann et al. 2016), where re-planning occurs frequently when new observations become available or sub-plans are expanded, can be sensitive to this issue – but it provides important features such as robustness and overall efficiency by dividing the overall planning problems in smaller chunks. A key observation in many robotic domains is that *robots repeatedly execute similar plans*. In this paper, we propose an approach and present initial results to mitigate the issue of long planning times by exploiting the repeating structure in robotic planning – that is, in the frequent execution of similar plans for different goals – through recording plans in a database and later extracting macros. The idea of task planning with macros, where typical action sequences are grouped into a macro action, has been used before in various ways to improve successor state generation by skipping intermediate states. Our approach combines several advantages, i.e., it determines macros a-priori off-line based on previous plans, it can employ and feed into several existing unmodified PDDL-based planners, and it creates domain-specific but problem-independent macros. Effectively, we trade off-line computation time for improved run-time planning performance. In robotics scenarios, this can be done either during robot idle times, or concurrently to execution. The overall performance can then be improved gradually in the long term.

Our approach operates on a plan database created a-priori by running planners on example problems. Distributing this computation with a cloud infrastructure allows for rapidly processing many instances. We carry out this first step off-line in order to accept longer planning times. If feasible, this could also be done concurrently to execution on-line. We then use MapReduce processing methods on a document-oriented and distributed plan database to identify macros of varying length with their frequency. For each macro, we generate a PDDL operator that can then again be used in a macro-augmented domain. For this macro operator, we determine the necessary preconditions based on the contained actions using regression and simplification, determine its effects through effect chaining, and coalesce parameters exploiting type information. Further domain executions will then allow estimating a macro's impact. The procedure can handle macros of arbitrary size (by incrementing the desired sequence length during identification) or may be be applied repeatedly to create hierarchical macros. If evaluation determines no positive or even a detrimental effect, macros may be pruned and excluded from further identification. By this, the speedup in planning improves over time until a sufficient domain-dependent number of plans have been analyzed.

We have evaluated the approach based on two classes of scenarios. We primarily focus on domains from the International Planning Competition (IPC). They provide a good basis for the general and extensive evaluation of macro generation. As a proof-of-concept robotics application domain, we use the logistics robot scenario of the upcoming robotics planning competition (Niemueller et al. 2016) that requires robots to transport workpieces among production machines. While the specific sequences are randomized and the costs vary, there are basic recurring sequences such as picking up a workpiece at one machine and delivering it to another.

Our contribution is the proposal of a novel method for *database-driven macro planning for STRIPS* (DBMP/S), that identifies macros from a database of recorded plans, and generates macros as PDDL operators with proper precondition and effects. We provide results on some IPC domains and a relevant robotics scenario, and a comparison to existing macro planning approaches. This paper is limited to the typed STRIPS fragment of PDDL. Ongoing work extends the approach to the ADL fragment.

In this paper, we provide some related work (Section 2) and then sketch our approach of data-driven macro planning in Section 3. The initial results we provide in Section 4 show that our approach outperforms planning without macros by at least an order of magnitude (given a suitable number of training samples, i.e., existing successful plans) and comparable or better performance than existing macro planning approaches, with a richer set of domain features and suitable planners. We conclude in Section 5.

2 Related Work

Planning deals with finding a sequence of actions to reach a given world state, called the goal state, from an initial state (Ghallab, Nau, and Traverso 2004). A world state is represented by a set of predicates. Planning operators are generalized actions which are described by their preconditions and effects. An action is a grounded planning operator. Depending on the formalism, certain restrictions apply to the representation of world states, preconditions, and effects.

PDDL Dialects. PDDL (McDermott et al. 1998) is the de-facto standard for defining planning problems and allows different formalisms depending on the given requirements. It supports STRIPS-like domains (Nilsson and Fikes 1972), but also more complex formalisms such as ADL (Pednault 1989) with types, disjunctive and quantified preconditions, and conditional effects.

Planners. Heuristic planners like FF (Hoffmann and Nebel 2001) and FASTDOWNWARD (Helmert 2006) use forward state space search with heuristics, e.g., FF relaxes the problem by ignoring delete effects. FASTDOWNWARD has a portfolio of heuristics. Both support ADL.

Macro Planning. In macro planning, a macro action is generated from a frequent or otherwise helpful action sequence. During search, the macro action and thus the action sequence is applied at once. Therefore, macro planning improves state space search by generating additional successor states which allows to skip intermediate states.

The planner STRIPS has been extended with macros in the form of *generalized plans* (Fikes, Hart, and Nilsson 1972), (partial) solutions to previous problems which are generalized by substituting constants with parameters. Generalized plans are domain-specific but problem-independent. The planner REFLECT (Dawson and Siklóssy 1977) is a STRIPS-based planner that generates problem-specific macro actions by analyzing the domain and the problem during the pre-processing phase. The Duet Planner (Gerevini et al. 2008) integrates domain-independent heuristic planning with domain-specific knowledge by combining the PDDL planner LPG (Gerevini, Saetti, and Serina 2003) with the HTN planner SHOP2 (Nau et al. 2003). In contrast to macro actions in other approaches, HTNs to be used by the Duet Planner must be specified manually by the domain designer.

Marvin (Coles and Smith 2007) is a FF-based macro planner that uses macros to escape plateaus during search. A plateau occurs if the heuristic value of all successor states is the same as or worse than the heuristic value of the current state. If Marvin encounters a plateau and escapes it, it remembers the escaping action sequence as macro. When the next plateau is encountered, macros from previous plateaus are considered as additional actions. Macros are only applied if there is a plateau. Marvin has been extended to store macro actions in a library so they can be re-used for solving subsequent problems (Coles, Fox, and Smith 2007). In order to keep the size of the library reasonable, macros are filtered based on usage count, number of problems since last use, instantiation count, and length of the macro action. Marvin's macro actions are planner-specific and cannot be re-used with other planners.

Wizard (Newton et al. 2008) learns STRIPS-based macros using a genetic algorithm. It generates initial macros from solutions of less complex seeding problems. It then applies a genetic algorithm to improve macros by prepending and appending actions, removing actions from the beginning and end, and splitting the macro into two. Wizard evaluates macros based on the percentage of problems solved, the mean time gain/loss, and the percentage of problems that were solved faster (Newton et al. 2007).

The planner MACROFF (Botea et al. 2005) uses two different approaches to macro planning: *Component Abstraction-Enhanced Domains* (CA-ED) and a *Solution-Enhanced Planner* (SOL-EP). With CA-ED the original domain is augmented with macro actions represented as normal PDDL actions. Thus, the planner can use macro actions as regular actions during search, the planner does not need to be adapted to use macros. CA-ED uses *component abstraction*, which combines low-level features linked by static facts into abstract components, which are then used to build macros. Macro actions are added to the original domain, no planner modification is necessary. CA-ED supports STRIPS domains. In contrast, SOL-EP does not represent macros as normal actions, but instead represents a macro as a partially ordered sequence of operators (Botea, Müller, and Schaeffer 2005), which can be applied in one step during search. Thus, the underlying planner needs to be modified. In SOL-EP, macros are generated from training problems and then statically filtered such that only the most promising macros are kept. SOL-EP supports ADL domains.

MUM (Chrpa, Vallati, and McCluskey 2014) is a STRIPS-based macro planner that learns macros from the solutions of less complex training problems and ranks and prunes macros based on the concept of *outer entanglements* (Chrpa and McCluskey 2012). Additionally, MUM uses *independent actions* (Chrpa 2010), i.e., actions that can be swapped without making the plan invalid, to generate macro actions from action sequences that are not consecutive in the original plans. In MUM, the planner is modified to consider outer entanglements of macros during search; thus, swapping the planner is not possible.

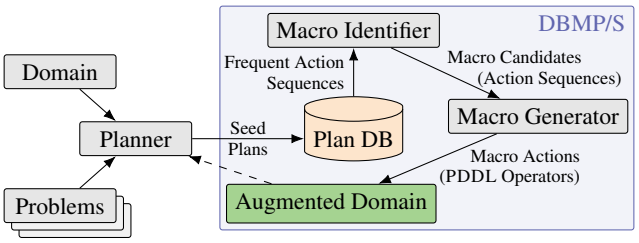

Figure 1: DBMP/S Architecture Overview

3 Database-Driven Macro Generation

The architecture of the proposed approach is shown in Figure 1. On the left, a classic PDDL setup consisting of domain definition, a number of problem instances, and a suitable planner is used to generate seed plans. These are stored in a plan database. This may be done on- or off-line, depending on the feasibility in the specific scenario. At suitable times, the macro identifier extracts frequent action sequences (other identification criteria may be used). The generator computes a PDDL operator with proper preconditions and effects. This operator is then integrated into an augmented domain which is used in subsequent planning. The augmented domain may be used for further (hierarchical) macro generation. Run-time problems are also stored in the database and may be used off-line later for non-hierarchical macro generation and evaluation.

Plan Database. Plans are stored in the document-oriented database MongoDB. It stores (nested) groups of key-value pairs called documents. Based on its use as a generic robot database (Niemueller, Lakemeyer, and Srinivasa 2012) this structure is useful as a unified storage for planning domains, problems, plans, and additional run-time information. Since the database is denormalized, that is, related data is embedded into a single document, queries processing can be scaled horizontally among many hosts easily.

Macro Identification. Potential macros are identified using the MapReduce (Dean and Ghemawat 2008) database query paradigm supported by MongoDB. An identification query then consists of two steps. First, a map function is applied to all plan documents of a specific domain. It emits one document consisting of an operator list and a parameter assignment for each sub-sequence of actions of a plan within specified length bounds, e.g., emit all sequences of length three to eight. For the parameter assignment, the map function always computes an assignment such that the total number of parameters is minimal, i.e., if the grounding of two operators in the sub-sequence have common values, then these parameters are represented as one parameter in the resulting macro candidate. Second, a reduce step then counts the occurrence of each sequence and parameter assignment. Action sequences are then selected for macro generation based on the sequence's frequency and its parameter reduction (other selection criteria are also possible).

Consider a plan database with the following plans (generated based on the actions shown in Listings 1 and 2):

```
Plan 1   1.  move-to-get(r1, C-RS, O, C-BS, O)
         2.  wp-get(r1, wp1, C-BS, O)
Plan 2   1.  move-to-get(r2, C-BS, O, C-CS1, O)
         2.  get(r2, wp2, C-CS1, O)
```

This yields the sequence <move-to-get,get>. Both plans assign the same values to some parameters of both actions. These parameters can be coalesced by merging the parameters ?to and ?mps to ?p4, and ?from-side, ?to-side, and ?side to ?p3 (cf. Listing 5).

Note that macro identification is decoupled from planning. Thus, we can scale the computation of seed plans in a cluster horizontally to parallelize.

Macro Generation. The goal of macro generation is to compute a PDDL action representation for an action sequence including a parameter assignment from the identification step. We restrict this work to STRIPS actions. To compute a STRIPS representation of an action sequence, the action's precondition and effects have to be identified.

For the *precondition*, we use a form of regression similar to (Newton et al. 2007). For a given action sequence $\langle a_1, a_2, \ldots, a_n \rangle$, we regress each precondition π_i of each action a_i over all effects $e_{i-1}, e_{i-2}, \ldots, e_1$ of the previous actions. To regress a precondition π on a single effect e, we proceed as follows[1]: (1) If π is a conjunction $\pi = \bigwedge_j \pi_j$, we regress all sub-formulae π_j to π'_j and obtain $\pi' = \bigwedge_j \pi'_j$, analogously for disjunctions. (2) If $\pi = P(\vec{s})$ and $e = P(\vec{t})$ for some predicate P, then we regress π to $\pi' = (\vec{s} = \vec{t} \vee P(\vec{s}))$. (3) Similarly, if $\pi = \neg P(\vec{s})$ and $e = P(\vec{t})$ for some predicate P, we regress π to $\pi' = (\vec{s} \neq \vec{t} \wedge \neg P(\vec{s}))$. (4) If $\pi = (\neg)P(\vec{s})$ and $e = Q(\vec{t})$ for some distinct predicates P and Q, the effect is unrelated to π and we get $\pi' = \pi$. For a negated effect $e = \neg P(\vec{t})$, we proceed analogously.

[1] $\vec{t}$ denotes a sequence of parameters $t_1, t_2, \ldots, t_k$

```
(:action move-to-get
 :parameters (?r - robot ?from - location
  ?from-side - mps-side ?to - mps ?to-side - mps-side)
 :precondition (and
  (entered-field ?r) (at ?r ?from ?from-side)
  (location-free ?to ?to-side) (can-hold ?r)
  (mps-state ?to READY-OUTPUT))
 :effect (and (not (at ?r ?from ?from-side))
  (at ?r ?to ?to-side) (location-free ?from ?from-side)
  (not (location-free ?to ?to-side))))
```

Listing 1: The action move-to-get

```
(:action wp-get
 :parameters (?r - robot ?wp - workpiece
              ?m - mps ?side - mps-side)
 :precondition (and (at ?r ?m ?side) (can-hold ?r)
  (wp-at ?wp ?m ?side) (wp-usable ?wp)
  (mps-state ?m READY-OUTPUT))
 :effect (and (not (wp-at ?wp ?m ?side))
  (holding ?r ?wp) (not (can-hold ?r))
  (not (mps-state ?m READY-OUTPUT))
  (mps-state ?m IDLE)))
```

Listing 2: The action wp-get

```
:effect (and
  (not (at ?p1 ?p2 ?p3)) (location-free ?p2 ?p3)
  (not (location-free ?p4 ?p3)) (at ?p1 ?p4 ?p3)))
```

Listing 3: The effect of `move-to-get` after reassignment.

```
:precondition (and (at ?p1 ?p4 ?p3) (can-hold ?p1)
  (wp-at ?p5 ?p4 ?p3) (wp-usable ?p5)
  (mps-state ?p4 READY-AT-OUTPUT))
```

Listing 4: The precondition of `wp-get` after reassignment.

(5) Finally, if e is a conjunction $e = \bigwedge_j e_j$, we regress π on each e_j successively. After regressing all preconditions π_i in this manner, we obtain a precondition π' as precondition of the macro. If π' holds, then all actions of the sequence can be executed consecutively.

Going back to the previous example, we need to regress the precondition π of the action `wp-get` on the effects e of `move-to-get`, i.e., following rule 1 and 5, we need to regress each conjunct of the precondition π shown in Listing 4 successively on all conjuncts of the effect e shown in Listing 3. Consider the conjunct `(at ?p1 ?p4 ?p3)` of π. First, regressing π on the sub-effect `(not (at ?p1 ?p2 ?p3))` of e yields `(and (not (= ?p2 ?4))(at ?p1 ?p4 ?p3))` by rule 3. The resulting conjunct `(at ?p1 ?p4 ?p3)` is further regressed to `true` because of the sub-effect `(at ?p1 ?p4 ?p3)` of e.

For the *effect*, we need to concatenate all effects while removing each effect conflict. For an action sequence $\langle a_1, a_2, \ldots, a_n \rangle$ and corresponding effects $\sigma = \langle e_1, e_2, \ldots, e_n \rangle$, we start with e_n and add each previous effect consecutively with effect chaining. To chain two effects e, f, where e occurs in σ before f, we proceed as follows: 1. If e is a conjunction $e = \bigwedge_i e_i$, we chain each e_i with f to e'_i and obtain $e' = \bigwedge_i e'_i$. 2. If $e = f$, we remove e. 3. If $e = \neg f$, we remove e. 4. If f is a conjunction $f = \bigwedge_i f_i$, we chain e with each f_i consecutively to e'_i and obtain $e' = \bigwedge_i e'_i$. 5. If $e = P(\vec{s})$ and $f = Q(\vec{t})$ for some distinct predicates P and Q, we keep e, i.e., $e' = e$. We do not allow the case $e = P(\vec{s})$, $f = \neg P(\vec{t})$ and the analogous case $e = \neg P(\vec{s})$, $f = P(\vec{t})$ because this would require a conditional effect to distinguish the cases $\vec{s} \neq \vec{t}$ and $\vec{s} = \vec{t}$.

```
(:action move-to-get_wp-get
  :parameters (?p1 - robot ?p2 - location
               ?p3 - mps-side ?p4 - mps ?p5 - workpiece)
  :precondition (and
   (entered-field ?p1) (at ?p1 ?p2 ?p3) (wp-usable ?p5)
   (location-free ?p4 ?p3) (mps-state ?p4 READY-OUTPUT)
   (can-hold ?p1) (not (= ?p2 ?p4)) (wp-at ?p5 ?p4 ?p3))
  :effect (and
   (not (wp-at ?p5 ?p4 ?p3)) (mps-state ?p4 IDLE)
   (not (mps-state ?p4 READY-OUTPUT))
   (at ?p1 ?p4 ?p3) (not (at ?p1 ?p2 ?p3))
   (holding ?p1 ?p5) (location-free ?p2 ?p3)
   (not (can-hold ?p1)) (not (location-free ?p4 ?p3))))
```

Listing 5: The macro `move-to-get,wp-get`

```
:effect (and (not (wp-at ?p5 ?p4 ?p3))
 (holding ?p1 ?p5) (not (can-hold ?p1))
 (not (mps-state ?p4 READY-OUTPUT))
 (mps-state ?p4 IDLE)))
```

Listing 6: The effect of `wp-get` after reassignment.

For the effect of the macro `move-to-get_wp-get`, we need to chain the effects in Listing 3 and Listing 6. In this case, the resulting effect is simply the conjunction of the two effects, as shown in Listing 5.

Macro Expansion. Actual task planning is performed using the augmented domain. Plans may therefore contain macros as an action. To execute the macro, it is expanded by replacing it with the original action sequence. Parameters are bound according to the macro instantiation (the specific assignment must be retrieved aside from the PDDL specification). Execution monitoring can therefore operate in the same way compared to the case without using macros.

Macro Evaluation. Once macros have been generated, they must be evaluated to determine whether they are useful. We perform *off-line evaluation* by adding (possibly multiple) macros to a domain and solving problems stored in the database with these macro-augmented domains. The result is then assessed with *evaluation metrics* such as mean planning time or expected execution time. Based on these metrics, the best macro configuration is selected. As an example, in the *Logistics Robots* domain, the best configuration was the domain with two macros with two actions each. In contrast to the other domains, adding one macro did not improve the planner performance to a great extent, which is why we decided to add both macros.

4 Initial Results

We evaluated DBMP/S based on three domains from the International Planning Competition (IPC): the Hiking and Barman domains from IPC-2014 and the Blocksworld domain from IPC-2011. For Hiking and Barman, we use the 20 original problems. For Blocksworld, we generated 100 random problem instances with 20 blocks with the problem generator provided by the competition. For the Logistics Robots scenario, we generated problems with 0 to 2 orders of each of the four product complexities resulting in 81 combinations. All instances were used for training and evaluation.[2]

Computing Setup. We used an in-house Kubernetes[3] cluster of 6 machines with a quad-core CPU Intel i7 and 16 GB RAM each. Planning systems are wrapped in Docker containers scheduled and distributed automatically among the cluster for execution. Domains, problems, and results are stored in MongoDB through an adapter program.

Seed Plans. Seed plans for the plan database were generated using FastDownward. We used a configuration that focuses on plan quality rather than speed to improve macro quality. For the same reason, it was not used during run-time

[2]The source code, data sets, and results are available at http://www.fawkesrobotics.org/p/dbmp-strips.

[3]Kubernetes website: http://kubernetes.io

evaluation. The planner was limited to 60 min run-time, 1 CPU thread, and 7 GB memory usage. This allowed us to run 12 planning processes concurrently. Computing seed plans required about 71 h processing time.

Macros. Macro candidates were identified for each domain. For this work, we limited the DBMP/S augmented domain to a single macro of length 2 for all domains but the *Logistics Robots* domain, where we used two macros of length 2.

Evaluation. FF and Marvin (without macro library) were used for the problem instances of the original as well as the augmented domain, MACROFF (SOL-EP) was run solely on the original domain. Planners were limited to 1 CPU thread, 4 GB memory, and 30 min run-time.

Results (Table 1). Already a single (frequently occurring) macro has a significant impact on solving capabilities and mean run-time. FF with DBMP/S found more solutions in shorter times. For example, on the Barman domain, where FF on the original domain could not find any solutions, it was able to solve 19 out of 20 problems with the augmented domain and solved most problem instances in less than 10 s. As the *Blocksworld* domain shows, using macros does not lead to significant overhead on simple problems. In the first quartile, FF with DBMP/S performs as well as FF alone. Compared to MACROFF and Marvin, database-driven macro planning shows comparable or even slightly better performance. In the Blocksworld domain, MACROFF solved most instances and in the shortest mean time. However, FF with the augmented domain provided better times on Q1 instances. The overhead of FF with DBMP/S is less than Marvin's in Q1.

	FF	MACROFF	Marvin	DBMP/S (FF)	DBMP/S (Marvin)
Blocksworld (100 problems)					
# solved	76	**100**	93	85	93
mean (s)	25.2	**0.72**	41.1	11.5	33.0
Q1 (s)	**0.006**	0.16	0.12	**0.006**	0.15
Q2 (s)	0.12	0.23	0.23	**0.017**	0.25
Q3 (s)	124	**0.40**	0.64	1.57	0.94
Hiking (20 problems)					
# solved	12	15	15	**19**	15
mean (s)	260	349	264	**196**	284
Q1 (s)	91.0	86.6	10.9	**5.67**	11.8
Q2 (s)	280	339	182	**25.6**	273
Q3 (s)	×	1751	1447	**268**	1653
Barman (20 problems)					
# solved	0	0	6	19	**20**
mean (s)	×	×	125	**4.60**	147
Q1 (s)	×	×	256	**2.10**	6.76
Q2 (s)	×	×	×	**2.96**	17.3
Q3 (s)	×	×	×	**5.77**	35.4
Logistics Robots (81 problems)					
# solved	4	†	3	4	3
mean (s)	0.156	×	0.74	**0.094**	0.40

Table 1: Benchmark results for FF, MACROFF, and Marvin without augmented domains, and FF and Marvin with DBMP/S domains augmented by a single macro of length 2 (limits: 1 CPU thread, 30 min run-time, and 4 GB RAM). The results are grouped into one section for each domain. The rows per section contain the number of problems solved, the mean time used on solved instances, and the respective run-time quartile, i.e., after sorting all solved instances by time ascending, Q1 represents the specific time of the instance at 25 % of the number of all problem instances, Q2 at 50 % (median), and Q3 at 75 %. The best value per row is indicated with bold face, × denotes no value and † a run-time error. In the Logistics Robots domain, no planner could solve 25 % of the problems, thus no quartile is given.

In the *Hiking* domain, FF with DBMP/S could solve the most problems with the shortest mean time. Marvin with DBMP/S performed slightly worse than without DBMP/S.

The *Barman* domain shows that on hard domains, FF with macros has a significant performance advantage compared to Marvin and MACROFF. In part, this is due to the fact that we were able to generate seed plans using FASTDOWNWARD. Thus, even though FF is not able to solve any problems in the domain, we are still able to generate macros, which can then be used by FF. Therefore, the ability to exchange planners to generate seed plans offers a critical advantage, which the other macro approaches cannot.

Finally, in the *Logistics Robots* domain, only the simplest problems with a single order could be solved at all. No macro planner could solve more problems than FF without macros. MACROFF failed with an error during macro extraction and Marvin could solve only 3 problems, both with and without DBMP/S macros. FF with DBMP/S macros solved 4 problems with the shortest mean time.

5 Conclusion

DBMP/S is a novel database-driven approach for macro planning for STRIPS domains formulated using PDDL. Collecting plans allows for long-term adaptation to domains that produce repeating action sequences in typical plans, as can be observed particularly in robotics domains. These action sequences are identified using the MapReduce paradigm supported by the document-oriented (plan) database MongoDB, allowing for a scalable way to determine the frequency (or other criteria) of a vast number of (sub-)sequences of operators quickly. Common parameters of actions in the sequence are coalesced to reduce the number of successor states generated for a macro. From such operator sequences, macros are formulated as normal PDDL actions with the proper preconditions and effects.

The approach was evaluated on the IPC domains Blocksworld, Hiking, and Barman, and the ICAPS Logistics Robots competition scenario. We compared DBMP/S to planning without macros and other macro planning approaches. The initial results show that DBMP/S improves planning performance significantly compared to planning without macros, even if we restrict the planner to use only a single macro containing two actions. In comparison to MACROFF and Marvin, DBMP/S shows similar and improved performance.

The key benefits of our approach are its improved planning times for domains where seed plans can be generated, its ability for long-term improvement of planning domains by collecting and analyzing a growing number of samples

from the database, and the ability to use (and exchange) unmodified PDDL planners for seeding or at run-time.

Acknowledgments

T. Hofmann and T. Niemueller are supported by the German National Science Foundation (DFG) research unit *FOR 1513* on Hybrid Reasoning for Intelligent Systems (http://www.hybrid-reasoning.org).

References

Botea, A.; Enzenberger, M.; Müller, M.; and Schaeffer, J. 2005. Macro-FF: Improving AI planning with automatically learned macro-operators. *Journal of Artificial Intelligence Research (JAIR)* 24.

Botea, A.; Müller, M.; and Schaeffer, J. 2005. Learning partial-order macros from solutions. In *Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS)*.

Brenner, M., and Nebel, B. 2009. Continual planning and acting in dynamic multiagent environments. *Autonomous Agents and Multi-Agent Systems* 19(3).

Chrpa, L., and McCluskey, T. L. 2012. On exploiting structures of classical planning problems: Generalizing entanglements. In *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI)*.

Chrpa, L.; Vallati, M.; and McCluskey, T. L. 2014. MUM: A Technique for Maximising the Utility of Macro-operators by Constrained Generation and Use. In *Proceedings of the 24th International Conference on Automated Planning and Scheduling*.

Chrpa, L. 2010. Generation of macro-operators via investigation of action dependencies in plans. *The Knowledge Engineering Review* 25(3).

Coles, A., and Smith, A. 2007. Marvin: A heuristic search planner with online macro-action learning. *Journal of Artificial Intelligence Research (JAIR)* 28.

Coles, A.; Fox, M.; and Smith, A. 2007. Online Identification of Useful Macro-Actions for Planning. *Artificial Intelligence*.

Dawson, C., and Siklóssy, L. 1977. The Role of Preprocessing in Problem Solving Systems. *Proceedings of the 5th International Joint Conference on Artificial Intelligence (IJCAI)*.

Dean, J., and Ghemawat, S. 2008. MapReduce: Simplified Data Processing on Large Clusters. *Comm. of the ACM* 51.

Fikes, R. E.; Hart, P. E.; and Nilsson, N. J. 1972. Learning and executing generalized robot plans. *Artificial Intelligence* 3.

Gerevini, A.; Kuter, U.; Nau, D.; Saetti, A.; and Waisbrot, N. 2008. Combining Domain-Independent Planning and HTN Planning: The Duet Planner. In *Proceedings of the 23rd Conference on Artificial Intelligence (AAAI)*.

Gerevini, A.; Saetti, A.; and Serina, I. 2003. Planning through stochastic local search and temporal action graphs in LPG. *Journal of Artificial Intelligence Research (JAIR)* 20.

Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated planning - theory and practice*. Morgan Kaufmann Publishers.

Helmert, M. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research (JAIR)* 26.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research (JAIR)* 14.

Hofmann, T.; Niemueller, T.; Claßen, J.; and Lakemeyer, G. 2016. Continual Planning in Golog. In *Proceedings of the 30th Conference on Artificial Intelligence (AAAI)*.

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. PDDL - The Planning Domain Definition Language.

Nau, D.; Au, T.-C.; Ilghami, O.; Kuter, U.; Murdock, J. W.; Wu, D.; and Yaman, F. 2003. SHOP2 : An HTN Planning System. *Journal of Artificial Intelligence Research (JAIR)* 20.

Newton, M. H.; Levine, J.; Fox, M.; and Long, D. 2007. Learning macro-actions for arbitrary planners and domains. In *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS)*.

Newton, M. H.; Levine, J.; Fox, M.; and Long, D. 2008. Wizard: Compiled Macro-Actions for Planner-Domain Pairs.

Niemueller, T.; Karpas, E.; Vaquero, T.; and Timmons, E. 2016. Planning Competition for Logistics Robots in Simulation. In *WS on Planning and Robotics (PlanRob) at Int. Conf. on Automated Planning and Scheduling (ICAPS)*.

Niemueller, T.; Lakemeyer, G.; and Srinivasa, S. 2012. A Generic Robot Database and its Application in Fault Analysis and Performance Evaluation. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*.

Nilsson, N. J., and Fikes, R. E. 1972. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial intelligence* 2.

Pednault, E. 1989. ADL: Exploring the Middle Ground Between STRIPS and the Situation Calculus. *KR*.