Determining Representativeness of Training Plans: A Case of Macro-operators

Lukáš Chrpa*†, Mauro Vallati‡
*Artificial Intelligence Center, Czech Technical University in Prague
†Faculty of Mathematics and Physics, Charles University in Prague
‡School of Computing and Engineering, University of Huddersfield
Email: chrpaluk@fel.cvut.cz, m.vallati@hud.ac.uk

***Abstract*—Most *learning for planning* approaches rely on analysis of training plans. This is especially the case for one of the best-known learning approach: the generation of macro-operators (macros). These plans, usually generated from a very limited set of training tasks, must provide a ground to extract useful knowledge that can be fruitfully exploited by planning engines. In that, training tasks have to be representative of the larger class of planning tasks on which planning engines will then be run. A pivotal question is how such a set of training tasks can be selected.**

To address this question, here we introduce a notion of *structural similarity* of plans. We conjecture that if a class of planning tasks presents structurally similar plans, then a small subset of these tasks is representative enough to learn the same knowledge (macros) as could be learnt from a larger set of tasks of the same class. We have tested our conjecture by focusing on two state-of-the-art macro generation approaches. Our large empirical analysis considering seven state-of-the-art planners, and fourteen benchmark domains from the International Planning Competition, generally confirms our conjecture which can be exploited for selecting small-yet-informative training sets of tasks.

I. Introduction

Automated Planning, which deals with finding a sequence of actions from a given initial state to a desired goal state, is one of the most prominent challenges of Artificial Intelligence [1]. For tackling this problem, many powerful planning engines based on various techniques have been developed, especially thanks to the International Planning Competitions (IPCs) that have been organized since 1998[1].

Complementary to developing planning engines, the *learning for planning* research area aims at learning useful knowledge about a class of planning tasks that can be exploited for improving the performance of planning engines. Examples of learning for planning include the configuration of portfolios, such as PbP [2] or IBACOP [3], that delivered outstanding performance in recent IPCs.

From a different perspective, the analysis of training plans for acquiring knowledge that provides additional guidance for planning engines is a pivotal part of the learning for planning research area. Such knowledge can be, for example, encoded into domain models, effectively reformulating them. Generally speaking, reformulation techniques aim at improving efficiency of generic planning engines. Examples of reformulation include action schema splitting [4], entanglements [5], and domain model configuration [6].

[1]http://www.icaps-conference.org/index.php/Main/Competitions

Perhaps the best-known reformulation technique is the generation of macro-operators (macros for short). Macros that encapsulate sequences of ordinary planning operators can be encoded as ordinary planning operators in the domain model, and thus can be exploited in a planner-independent fashion [7], [8], [9]. Most of the macro learning techniques rely on a training phase in which they analyse a set of training plans. The number of required training plans considerably varies; for example, DBMP/S [10] uses 20–100 training plans while MUM [9] uses 5–8 training plans. One might ask whether, for example, MUM would have generated different macros if more training plans were used. Chrpa et al. [11] performed an empirical analysis on how different entanglements, relations between operators and predicates [5], are learnt from different numbers of training plans. To fruitfully exploit macros –as well as other reformulation approaches– on a class of planning tasks of interest, the planning tasks used for training purposes must be representative of such a class.

In this paper, we define a notion of *structural similarity of plans* that is based on the analysis of relations of predicate achievement between planning operators. We then exploit this notion to determine how structurally similar plans affect knowledge acquisition. In particular, focusing on macros, we investigate whether –in presence of structurally similar plans– a small set of training tasks is enough for deriving the relevant additional knowledge for the whole class of planning tasks. In a nutshell, we want to tackle the following question: *is it worth spending a considerable amount of CPU-time to train on a large set of training tasks, or a small training set is enough to generate the same macros?* In order to address such a question, we conducted a large empirical analysis considering two state-of-the-art macro generation approaches – MUM [9] and BloMA [12] –, seven state-of-the-art planners and all benchmark domains from the learning tracks of the IPC-6 and IPC-7.

II. Preliminaries

Classical planning is a restricted form of Automated Planning, where the environment is static and fully observable, and actions are deterministic and instantaneous [1]. In the classical (STRIPS) representation the environment is described by first-order logic *predicates*. *States* are defined as sets of grounded predicates (or atoms). We say that $o = (\mathit{name}(o),\ \mathit{pre}(o), \mathit{eff}^-(o), \mathit{eff}^+(o))$ is a *planning operator*, where $\mathit{name}(o) = \mathit{op\_name}(x_1, \ldots, x_k)$ ($\mathit{op\_name}$ is an unique operator name and $x_1, \ldots x_k$ are variable symbols (arguments) appearing in the operator) and $\mathit{pre}(o), \mathit{eff}^-(o)$ and $\mathit{eff}^+(o)$ are sets of (ungrounded) predicates with variables taken only from $x_1, \ldots x_k$ representing o's precondition, negative (delete), and positive (add) effects respectively. *Actions* are grounded instances of planning operators. An action a is *applicable* in a state s if and only if $\mathit{pre}(a) \subseteq s$. Application of a in s (if possible) results in a state $(s \setminus \mathit{eff}^-(a)) \cup \mathit{eff}^+(a)$.

A *planning domain model* is specified by a set of predicates and a set of planning operators. A *planning problem* is specified via a set of objects (used to instantiate predicates and operators), an initial state and a goal represented by a set of atoms. A *planning task* is specified by a domain model and a planning problem. Given a planning task, a *plan* is a sequence of actions such that their consecutive application starting in the initial state results in a state containing all the atoms representing the goal.

In every plan, every atom in a precondition of an action a_j is (necessarily) achieved in the sense that there exists an action a_i achieving the atom before a_j, or the atom is present in the initial state, and no other action in between a_i (or the initial state) and a_j does not delete or (re)achieve the atom. Notice that being an achiever relates to the notion of *causal link* in plan-space planning.

Definition 1. *Let $\pi = \langle a_1, a_2, \ldots a_n\rangle$ be a plan for a given planning task, where I is the initial state. We say that an action a_i (resp. the initial state I) is an* ***achiever*** *of an atom p for an action a_j in the plan π if and only if $i < j$, $p \notin \mathit{eff}^-(a_k) \cup \mathit{eff}^+(a_k)$ for every $k \in \{i+1, \ldots, j-1\}$ (resp. $k \in \{1, \ldots, j-1\}$), $p \in \mathit{eff}^+(a_i) \cap \mathit{pre}(a_j)$ (resp. $p \in I \cap \mathit{pre}(a_j)$).*

Consequently, for each action in a given plan and for each atom in its precondition it is the case that there is exactly one achiever of that atom. On the other hand, an action in a given plan might be an achiever of the same atom for more actions.

For getting a complete picture about which actions (or atoms available in the initial state) achieve all atoms required by another action we have to extend the previous definition as follows.

Definition 2. *Let $\pi = \langle a_1, a_2, \ldots a_n\rangle$ be a plan for a given planning task, where I is the initial state. Let $\mathcal{A}^\pi_j = \{\langle p, x\rangle \mid x \text{ is an achiever of } p \text{ for } a_j\}$ be a set of achievers for a_j. We say that $\mathcal{A}^\pi_j$ is a* ***complete set of achievers*** *for a_j in π if and only if $\mathit{pre}(a_j) = \{p \mid \langle p, x\rangle \in \mathcal{A}^\pi_j\}$.*

III. Structural Similarity of Plans

Determining the achiever relations in plans gives insights into how these plans are structured. Roughly speaking, we can identify which actions have to be applied in order to achieve preconditions of another action. However, analysing specific actions –that contain information about specific objects of the considered planning task– might result knowledge that is too focused on the given task, and that does not generalize on other planning tasks of the same domain model.

In order to analyse plans in domain-specific way rather than problem-specific, we have to generalize the achiever relations for planning operators.

Definition 3. *Let $\pi = \langle a_1, a_2, \ldots a_n\rangle$ be a plan for a planning task P, O be a set of planning operators defined in the domain model of P and I be the initial state of P. We say that $o_i \in O$ (resp. I) is an* ***achiever*** *of a predicate p for $o_j \in O$ in π if and only if for $a_j \in \pi$ being an instance of o_j, there exists $a_i \in \pi$ being an instance of o_i (resp. I) which is an achiever of a corresponding instance of p for a_j.*

In the generalised case, however, an operator can have more achievers of one predicate. This implies that an operator can have more than one complete set of achievers.

Definition 4. *Let $\pi = \langle a_1, a_2, \ldots a_n\rangle$ be a plan for a planning task P, O be a set of planning operators defined in the domain model of P and I be the initial state of P. Let $\mathcal{O}^\pi_j = \{\Phi^\pi_{j_1}, \ldots, \Phi^\pi_{j_k}\}$ be a set of elementary sets of achievers for $o_j \in O$ in π. An* ***elementary set of achievers*** *$\Phi^\pi_{j_i}$ for o_j in π is defined as $\Phi^\pi_{j_i} = \{\langle p, x\rangle \mid \langle p_{gnd}, x_{gnd}\rangle \in \mathcal{A}^\pi_{j_i}$ where p_{gnd}, x_{gnd} are instances of p, x respectively$\}$, where $\mathcal{A}^\pi_{j_i}$ is a complete set of achievers for $a_i \in \pi$ being an instance of o_j.*
$\mathcal{O}^\pi_j$ is ***complete set of achievers*** *for $o_j \in O$ in π if and only if for every a_i being an instance of o_j: $\Phi^\pi_{j_i} \in \mathcal{O}^\pi_j$*

Definition 4 indicates how we can determine structural similarity of plans that are generated for different planning tasks sharing the same domain model. Technically speaking, we can compare complete sets of achievers for every operator defined in the domain model, to determine whether two (or more) plans are structurally similar. Intuitively, if two plans have the same complete sets of achievers for every operator, then they are structurally similar. However, an initial state is a usual achiever of atoms for actions that are placed early in the plan, while for actions that are placed lately in the plan an initial state is an occasional achiever of atoms. This observation is also reflected in complete sets of achievers for planning operators. For example, a simple drive operator that has only one predicate (at ?car ?loc) in its precondition can have two achievers of that predicate – an initial state or the drive operator (i.e., itself). For toy planning tasks that have short plans, it might be the case that only an initial state is an achiever of (at ?car ?loc). Therefore, considering strict comparison of complete sets of achievers might determine some plans structurally different even if they might not be.

To mitigate the aforementioned issue we will consider initial states as "joker cards", i.e., if an initial state is an achiever of a predicate for an operator, we consider it to be equal to any other operator being in the same achiever relation (the same predicate and the same operator). Given this "trick", we can then define *structural similarity* of elementary sets of achievers, complete sets of achievers and plans.

Definition 5. *Let π and π' be plans for some planning tasks that share the same domain model. Let O be the set of planning operators defined in the domain model. Let $\Phi^{\pi}_{j_k}$ and $\Phi^{\pi'}_{j_l}$ be elementary sets of achievers for $o_j \in O$ in π and π' respectively. If for every $p \in pre(o)$ there exist $\langle p, x\rangle \in \Phi^{\pi}_{j_k}$ and $\langle p, x'\rangle \in \Phi^{\pi'}_{j_l}$ such that i) x or x' (or both) represents the initial state of the corresponding planning task, or ii) $x = x'$ (i.e., both represent the same operator), then $\Phi^{\pi}_{j_k}$ and $\Phi^{\pi'}_{j_l}$ are **structurally similar**.*
*Let $\mathcal{O}^{\pi}_j$ and $\mathcal{O}^{\pi'}_j$ be complete sets of achievers for $o_j \in O$ in π and π' respectively. If for every $\Phi^{\pi}_{j_k} \in \mathcal{O}^{\pi}_j$ there exists $\Phi^{\pi'}_{j_l} \in \mathcal{O}^{\pi'}_j$ such that $\Phi^{\pi}_{j_k}$ and $\Phi^{\pi'}_{j_l}$ are structurally similar, and for every $\Phi^{\pi'}_{j_l} \in \mathcal{O}^{\pi'}_j$ there exists $\Phi^{\pi}_{j_k} \in \mathcal{O}^{\pi}_j$ such that $\Phi^{\pi}_{j_k}$ and $\Phi^{\pi'}_{j_l}$ are also structurally similar, then $\mathcal{O}^{\pi}_j$ and $\mathcal{O}^{\pi'}_j$ are **structurally similar**.*
*If for every $o \in O$ it is the case that its complete sets of achievers $\mathcal{O}^{\pi}$ and $\mathcal{O}^{\pi'}$ are structurally similar, or $\mathcal{O}^{\pi} = \emptyset$ or $\mathcal{O}^{\pi'} = \emptyset$, then π and π' are **structurally similar**.*

We are specifically interested in determining whether plans for a set of planning tasks using the same domain model are generally structurally similar. Hence we define a notion of z-divergence that represents the number of plans (z) that are structurally different than the rest of the plans.

Definition 6. *Let $\Pi = \{\pi_1, \dots, \pi_n\}$ be a set of plans for planning tasks sharing the same domain model. Let $C_1, \dots, C_k$ be sets of plans such that i) $\bigcup_{i=1}^{k} C_i = \Pi$ and $C_i \cap C_j = \emptyset$ for every $i \neq j$, ii) for every C_i ($1 \leq i \leq k$) and $\pi_p, \pi_q \in C_i$, π_p and π_q are structurally similar, and iii) for every C_i ($1 \leq i \leq k$) and $\pi_p \in C_i$ there is no $\pi_q \in C_j$ ($i \neq j$) such that π_p and π_q are structurally similar. Then, we say that $C_1, \dots, C_k$ **classify** Π.*
*We also say that Π is **z-divergent**, where $z = \min_{i=1}^{k}(n - |C_i|)$.*

The meaning of z-divergence can be understood as a degree of structural similarity of plans in a given set. z ranges from 0 (the most similar) to $n-1$ (the most different).

In some cases, we can determine 0-divergence for any (non-empty) set of plans for planning tasks where the domain model is defined such that every predicate can be achieved by at most one planning operator.

Theorem 1. *Let Π be a non-empty set of plans for planning tasks sharing the same domain model. Let P be the set of predicates and O be the set of operators defined in the domain model. If for every $p \in P$ it is the case that there exists at most one operator $o \in O$ such that $p \in \mathit{eff}^{+}(o)$, then Π is 0-divergent.*

Proof Sketch. For every operator $o \in O$, an elementary set of achievers consists of couples $\langle p, x\rangle$, where x can represent either the initial state, or the only operator that achieves p. Hence, the 0-divergence of Π immediately implies from Definitions 5 and 6. □

As mentioned before z-divergence refers to how structurally similar plans are in a given set. In other words, z-divergence indicates how similar of different information the plans carry. With z close to 0 on n plans we can expect that its smaller subset of m plans carries the same information. Following this intuition we formulate a conjecture, focusing on macro learning techniques, as follows.

Conjecture 1. *Let Π be a set of plans such that for its z-divergence it is the case that $z \to 0$. Then for an arbitrary Π' such that $\Pi' \subset \Pi$, $|\Pi'| \ll |\Pi|$ and Π' is 0-divergent it is the case that macros generated from Π is the same as macros generated from Π'.*

The meaning of Conjecture 1 is that for a set of structurally similar plans we can select its (much smaller) subset for training purposes. However, it still requires to generate all plans which might be computationally expensive. On the other hand, we can expect that if we initially generate m training plans that are 0-divergent, then their (much lager) superset of n plans is z-divergent with z close to 0.

Conjecture 2. *Let Π be a set of plans that is 0-divergent. Then for an arbitrary Π' such that $\Pi' \supset \Pi$, $|\Pi'| \gg |\Pi|$ it is the case that Π' is z-divergent with $z \to 0$.*

IV. Empirical Evaluation

For verifying our conjectures, we considered two state-of-the-art macro learning techniques, MUM [9] and BloMa [12]. In the BloMa case, we considered the first phase of the learning process, generating "Extended Blocks" that are used as candidates for macros. This is because the candidates are then evaluated on how planners exploit them in order to be considered as macro, which is not an information that can be acquired from training plans. For generating training plans we selected seven state-of-the-art planners, according to exploited planning techniques and performance in recent competitions: *LPG-td* [13], *Lama* [14], *Probe* [15], *MpC* [16], *Yahsp3* [17], *Mercury* [18], and *Jasper* [19].

We included the benchmark domains used in the learning tracks of the 6th and 7th editions of the International Planning Competitions. Out of the selected domains, n-puzzle, Gripper and Spanner satisfy the conditions of Theorem 1. For each domain, we generated 20 tasks using available problem generators (for the IPC-6, training tasks were selected among those provided by the organizers). Training tasks were selected such that training plan lengths were in the order of tens of actions. Every planner was run on all training tasks, with a cutoff time of 120 seconds. Configurations (planner, domain) in which less than 10 training plans were generated were excluded from our analysis.

To verify Conjecture 1, for each couple (domain,planner) we selected, from the whole set of training plans, 5 training plans –solutions of the smallest considered training tasks– such that they were 0-divergent. If 0-divergent subset of the plans could not be selected, we relaxed the 0-divergence constraint and selected 5 plans, solutions of the smallest training tasks. The "close to zero" threshold for z-divergence was set to 2,

	LPG				MpC				Probe				Yahsp				LAMA				Mercury				Jasper			
	P	Z	M	B	P	Z	M	B	P	Z	M	B	P	Z	M	B	P	Z	M	B	P	Z	M	B	P	Z	M	B
Barman	12	10	✓	✓	14	11	✓	X	20	10	✓	X	10	8	✓	✓	20	11	✓	X	15	6	✓	✓	20	16	✓	✓
Bw	20	18	X	X	20	18	X	✓	20	18	X	X	20	18	✓	X	20	18	✓	✓	20	17	✓	✓	20	18	✓	✓
Depots	18	10	✓	X	17	9	✓	X	18	6	✓	X	15	9	✓	X	10	7	✓	X	9	6	-	-	15	6	✓	X
Gold-miner	20	9	X	✓	20	4	X	✓	20	0	✓	✓	19	11	X	X	20	4	✓	✓	19	10	X	✓	20	11	✓	✓
Gripper	18	0	✓	✓	16	0	✓	✓	12	0	✓	✓	8	0	-	-	11	0	✓	X	10	0	✓	✓	12	0	✓	✓
Matching-Bw	20	19	X	✓	18	17	X	✓	20	19	X	✓	20	19	X	X	20	19	X	✓	20	19	X	✓	20	19	X	✓
n-puzzle	20	0	✓	✓	1	0	-	-	19	0	✓	✓	18	0	✓	✓	20	0	✓	✓	20	0	✓	X	20	0	✓	✓
Parking	14	12	✓	✓	19	17	✓	X	19	17	✓	✓	20	19	✓	✓	19	18	✓	✓	19	18	✓	✓	19	17	✓	✓
Rovers	20	19	X	X	20	19	✓	✓	20	19	✓	✓	20	19	✓	X	20	19	X	✓	20	19	✓	X	20	19	X	X
Satellite	20	2	✓	✓	20	9	✓	✓	19	1	X	✓	20	4	✓	✓	20	1	✓	✓	20	1	✓	✓	20	2	X	✓
Sokoban	20	0	✓	✓	20	0	✓	✓	20	0	✓	✓	20	0	✓	✓	20	0	✓	✓	20	0	✓	✓	20	0	✓	✓
Spanner	20	0	X	✓	20	0	✓	X	12	0	✓	✓	12	0	✓	✓	12	0	✓	✓	12	0	X	✓	14	0	✓	✓
Thoughtful	13	12	X	X	17	16	X	X	19	18	X	✓	17	16	X	✓	19	18	X	✓	19	18	X	✓	20	19	X	✓
TPP	16	4	✓	✓	19	9	✓	✓	20	0	✓	✓	20	0	✓	✓	20	0	✓	✓	20	0	✓	✓	20	0	-	✓

TABLE I
THE RESULTS SHOWING THE NUMBER OF CONSIDERED TRAINING (P)LANS, (Z)-DIVERGENCE OF THE SETS OF THE PLANS, AND WHETHER LEARNT MACROS BY (M)UM AND EXTENDED (B)LOCK DECOMPOSITION FROM A SELECTION OF PLANS ARE THE SAME AS FOR THE WHOLE SET OF PLANS (✓ - IF YES, X - IF NOT). "-" DENOTES CASES WHERE NOT ENOUGH TRAINING PLANS WERE AVAILABLE, OR THE LEARNING METHOD FAILED.

i.e., 2-divergent sets of plans were considered as structurally similar.

Table I shows the results of our empirical analysis, in particular with regards to the fact that the set of extracted macros was the same for the whole set of training plans, as well as for the selection of 5 training plans (from the corresponding set).

Out of the considered domains, seven have at least a set of structurally similar training plans. Gripper, n-puzzle, Spanner and Sokoban have 0-divergent sets of training plans for every considered planner. For Gripper, n-puzzle, and Spanner, the 0-divergence implies from Theorem 1. In the remaining domains, TPP has five 0-divergent sets of training plans, Gold-miner one, and Satellite has five sets of training plans whose z-divergence is "close to zero".

Summarising, there are 37 cases, configurations (planner, domain), where structural similarity holds (i.e., the corresponding set of training plans is at most 2-divergent), and 51 configurations in which structural similarity does not hold (i.e., the corresponding set of training plans is more than 2-divergent). By considering the 2 learning techniques, we have 73 comparisons for the structurally similar configurations[2] and 102 comparisons for the "structurally different" configurations. In 66 out of 73 "structurally similar" comparisons, the same macros were extracted for the whole training plan sets as well as for its 5 plan subsets. Instead, in only 58 out of 102 "structurally different" comparisons the same macros were extracted for the whole training plan sets as well as for its 5 plan subsets. That means that in 92% of the cases when structurally similar plans were considered the same macros can be extracted by considering a very small set of training instances. In contrast, the same applies to only about 57% of the other cases. The results hence support Conjecture 1.

To verify Conjecture 2, we have considered, for each configuration (domain,planner), 5 training plans, solutions of the simplest tasks in the set. If the considered set of plans was 0-divergent, then we considered the whole set of plans to be structurally similar (i.e., at most 2-divergent). Otherwise, the whole set of plans was considered as "structurally different" (i.e., more than 2-divergent). Out of 98 configurations, in only one case (Depots,probe) we did a wrong classification (the considered 5 plans were 0-divergent while the whole set of plans was 6-divergent and thus "structurally different"). Hence, in total, the success rate was nearly 99% which supports Conjecture 2.

[2]MUM had crashed for Jasper's plans in TPP and thus is excluded

A. Discussion

The results support our conjectures and, in the larger context, indicate that by exploiting the structural similarity of plans, using a small number of training plans results in generating the same knowledge (macros) as for larger sets of training plans and thus we can save computational efforts required to generate a large set of training plans as well as processing these plans by (macro) learning techniques (e.g. MUM). Hence, we can initially generate a small number of training tasks (e.g. 5) and solve them by a given planner. If the training plans are 0-divergent, we use only these plans (and tasks) in the (macro) learning process since they are, with high confidence, representative enough. Otherwise, the plans might not be representative enough and we might need to generate a larger set of training tasks (and plans).

The results also go along the line with Chrpa et al.'s work on how different entanglements can be learnt from differently large sets of training plans [11]. In domains where we identified that plans are always or very often structurally similar, namely Gripper, Satellite Spanner and TPP, they reported that there is no (or very small) difference on learnt entanglements. On the other hand, in domains where we identified that plans tend to be structurally different, namely Barman, Bw, Depots, Matching-Bw, Parking, Rovers and Thoughtful, they reported (except Bw and Parking) differences on learnt entanglements.

Approaches such as Macro-FF [8], Wizard [20] or BloMa [12] incorporate a validation phase, where planners are run on domains enhanced by macro candidates and macros that are used frequently or demonstrate planner's performance increase are kept while the others are discarded. Structural similarity of plans, on the other hand, plays a role in knowledge acquisition that is performed solely on the training plans and might not have effect on the validation phase. As we have shown on the BloMa example, structural similarity of plans can help with a part of the training which relies on analysing training plans and generating preliminary knowledge that is further processed in the validation phase.

V. Conclusions

In this paper, we defined a notion of structural similarity of plans based on the relation of predicate achievement between planning operators. We then raised two conjectures stating that if a small set of generated training plans is structurally similar, then such a set yields (very likely) the same knowledge (macros) as could be acquired by exploring a larger set of training plans.

Our empirical analysis, focused on two techniques for generating macro-operators, demonstrated that our conjectures are generally confirmed and that the introduced notion of similarity can provide a fruitful support for learning for planning approaches that analyse training plans. Consequently, for classes of planning tasks with structurally similar plans only a very small number of training plans is needed to acquire reasonably representative knowledge, with a clear reduction in terms of computational resources for generating training plans as well as performing the training (of macros) itself.

In future, we would like to investigate how structural information about plans can be utilised in learning domain control knowledge, or heuristics. In a nutshell, we believe that from plan structure it can be possible to extract knowledge such as extra preconditions for operators that would prune the search space. Alternatively, structural knowledge of plans can be used to prioritize specific planning operators in given states (a similar idea has been exploited by Roller [21]). We are also interested in studying whether we can assess and evaluate structural similarity of plans by exploiting planning features [22] or tools such as Torchlight [23] that can be useful in predicting performance of planning engines. Finally, we believe that the notion of structural similarity can be fruitfully exploited in case-based planning [24], [25].

Acknowledgements

This research was funded by the Czech Science Foundation (project no. 18-07252S) and by the OP VVV funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics".

References

[1] M. Ghallab, D. Nau, and P. Traverso, *Automated planning, theory and practice*. Morgan Kaufmann Publishers, 2004.

[2] A. Gerevini, A. Saetti, and M. Vallati, "Planning through automatic portfolio configuration: The pbp approach," *J. Artif. Intell. Res. (JAIR)*, vol. 50, pp. 639–696, 2014.

[3] I. Cenamor, T. de la Rosa, and F. Fernández, "The ibacop planning system: Instance-based configured portfolios," *J. Artif. Intell. Res. (JAIR)*, vol. 56, pp. 657–691, 2016.

[4] C. Areces, F. Bustos, M. Dominguez, and J. Hoffmann, "Optimizing planning domains by automatic action schema splitting," in *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, (ICAPS)*, 2014.

[5] L. Chrpa and T. L. McCluskey, "On exploiting structures of classical planning problems: Generalizing entanglements," in *ECAI*, 2012, pp. 240–245.

[6] M. Vallati, F. Hutter, L. Chrpa, and T. L. McCluskey, "On the effective configuration of planning domain models," in *Proceedings of the International Joint Conference on AI (IJCAI)*, 2015, pp. 1704–1711.

[7] R. Korf, "Macro-operators: A weak method for learning," *Artificial Intelligence*, vol. 26, no. 1, pp. 35–77, 1985.

[8] A. Botea, M. Enzenberger, M. Müller, and J. Schaeffer, "Macro-ff: Improving ai planning with automatically learned macro-operators," *Journal of Artificial Intelligence Research (JAIR)*, vol. 24, pp. 581–621, 2005.

[9] L. Chrpa, M. Vallati, and T. L. McCluskey, "MUM: a technique for maximising the utility of macro-operators by constrained generation and use," in *ICAPS*, 2014, pp. 65–73.

[10] T. Hofmann, T. Niemueller, and G. Lakemeyer, "Initial results on generating macro actions from a plan database for planning on autonomous mobile robots," in *ICAPS*, 2017, pp. 498–503. [Online]. Available: https://aaai.org/ocs/index.php/ICAPS/ICAPS17/paper/view/15762

[11] L. Chrpa, M. Vallati, and H. Osborne, "Learnability of specific structural patterns of planning problems," in *ICTAI*, 2013, pp. 18–23.

[12] L. Chrpa and F. H. Siddiqui, "Exploiting block deordering for improving planners efficiency," in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, 2015, pp. 1537–1543. [Online]. Available: http://ijcai.org/Abstract/15/220

[13] A. Gerevini, A. Saetti, and I. Serina, "Planning through stochastic local search and temporal action graphs," *Journal of Artificial Intelligence Research (JAIR)*, vol. 20, pp. 239 – 290, 2003.

[14] S. Richter and M. Westphal, "The lama planner: guiding cost-based anytime planning with landmarks," *Journal Artificial Intelligence Research (JAIR)*, vol. 39, pp. 127–177, 2010.

[15] N. Lipovetzky, M. Ramirez, C. Muise, and H. Geffner, "Width and inference based planners: Siw, bfs(f), and probe," in *The Eighth International Planning Competition. Description of Participant Planners of the Deterministic Track*, 2014, pp. 6–7.

[16] J. Rintanen, "Madagascar: Scalable planning with sat," in *The Eighth International Planning Competition. Description of Participant Planners of the Deterministic Track*, 2014, pp. 66–70.

[17] V. Vidal, "Yahsp3 and yahsp3-mt in the 8th international planning competition," in *The Eighth International Planning Competition. Description of Participant Planners of the Deterministic Track*, 2014, pp. 64–65.

[18] C. Domshlak, J. Hoffmann, and M. Katz, "Red-black planning: A new systematic approach to partial delete relaxation," *Artif. Intell.*, vol. 221, pp. 73–114, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.artint.2014.12.008

[19] F. Xie, M. Müller, and R. Holte, "Jasper: the art of exploration in greedy best first search," in *The Eighth International Planning Competition. Description of Participant Planners of the Deterministic Track*, 2014.

[20] M. A. H. Newton, J. Levine, M. Fox, and D. Long, "Learning macro-actions for arbitrary planners and domains," in *ICAPS 2007*, 2007, pp. 256–263.

[21] T. de la Rosa, S. J. Celorrio, R. Fuentetaja, and D. Borrajo, "Scaling up heuristic planning with relational decision trees," *J. Artif. Intell. Res. (JAIR)*, vol. 40, pp. 767–813, 2011.

[22] C. Fawcett, M. Vallati, F. Hutter, J. Hoffmann, H. H. Hoos, and K. Leyton-Brown, "Improved features for runtime prediction of domain-independent planners," in *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS*, 2014.

[23] J. Hoffmann, "Analyzing search topology without running any search: On the connection between causal graphs and h+," *Journal of Artificial Intelligence Research*, vol. 41, pp. 155–229, 2011.

[24] D. Borrajo, A. Roubíčková, and I. Serina, "Progress in case-based planning," *ACM Computing Surveys (CSUR)*, vol. 47, no. 2, p. 35, 2015.

[25] L. Spalazzi, "A survey on case-based planning," *Artificial Intelligence Review*, vol. 16, no. 1, pp. 3–36, 2001.