

# Combining Learning Techniques for Classical Planning: Macro-operators and Entanglements

Lukáš Chrpa  
Agent Technology Center  
Faculty of Electrical Engineering  
Czech Technical University in Prague  
chrpa@agents.felk.cvut.cz

*Abstract*—Planning techniques recorded a significant progress during recent years. However, many planning problems remain still hard even for modern planners. One of the most promising approaches is gathering additional knowledge by using learning techniques. Well known sort of knowledge - macro-operators, formalized like ‘normal’ planning operators, represent a sequence of primitive planning operators. The other sort of knowledge consists of pruning unnecessary operators’ instances (actions) by investigating connections (entanglements) between operators and initial or goal predicates. Advantageously, macro-operators and entanglements can be encoded directly in planning domains (or problems) and common planning systems can be applied on them. In this paper, we will show how we can put these approaches together. We will provide an experimental evaluation showing that combining these learning techniques can improve the planning process.

## I. INTRODUCTION

Planning problems [1] belong to one of the most important research areas of Artificial Intelligence (AI). Thanks to the International Planning Competition (IPC)<sup>1</sup>, many advanced planning systems have been developed. Neo-classical planning techniques explore the Planning Graph [2], a structure that helps to avoid symmetries usually appearing during the ‘classical’ forward or backward search. Successful planning systems (like [3], [4]) are based on heuristics search [5], where heuristics are usually computed over the (relaxed) Planning Graph. Some planning systems also benefit from a translation of planning problems into other formalisms like Boolean Satisfiability (SAT) [6] or Constraint Satisfaction Problems (CSP) [7].

However, complexity of (classical) planning is up to EXPSPACE-complete [8] (considering the classical representation of planning problems [1]). It causes that even the best planning systems often fail to solve little harder planning problems. Despite the existence of many powerful heuristics planning itself is still mainly about ‘brute force’ search. This is the main reason why many researchers are starting to focus on using learning techniques that can gather additional knowledge describing some characteristics for particular classes of planning problems to support the planning process. This knowledge can be, for instance, learnt from training plans - solutions of simpler planning problems of the given class.

Generating macro-operators [9] is well known and thoroughly studied sort of knowledge. Macro-operators can be understood as ‘normal’ planning operators representing a sequence of primitive planning operators. There have been developed several approaches [10], [11], [12], [13], [14] that can generate macro-operators. Macro-operators serve like ‘shortcuts’ in the search space that reduce its depth. On the other hand, macro-operators are responsible for an increase of the branching factor. Additionally, in [13], [14] all primitive operators replaced by newly generated macro-operators in all training plans are removed. It gained very promising results, especially the planners’ running times were significantly reduced.

Eliminating unnecessary actions is a possibility how the branching factor can be reduced. Work presented in [15], [14] learns connections (called entanglements) between planning operators and initial or goal atoms (predicates). Actions (normally reachable) that violate the learned entanglements are eliminated. This method gained very promising results as well.

In this paper, we are focusing on a combination of the both learning techniques (macro-operators and entanglements). We believe that it can help with reducing both the depth of the search space and the branching factor. We will experimentally evaluate how the planning process can be improved when macro-operators and entanglements are used together.

This paper is organized as follows. First, we will introduce basic notions from the planning theory followed by a brief discussion about related works, then we will say what macro-operators and entanglements stand for. Then, we will formally show how macro-operators and entanglements can be combined. It is followed by an example which will help the reader to understand how it works. Then, we will provide a thorough experimental evaluation and then we will finally conclude.

## II. PRELIMINARIES

Traditionally, AI planning deals with the problem of finding a sequence of actions transforming the world from some initial state to a goal state. A **predicate**  $p(t_1, \dots, t_n)$  is a construct, where  $p$  represents a predicate symbol and  $t_1, \dots, t_n$  are terms (variables or constants). A predicate is **grounded** if and only if all its terms are constants. A **state**  $s$  is a set of (grounded) predicates that are true in  $s$ . An **Action**  $a$  is a triple  $(pre(a), eff^-(a), eff^+(a))$  of sets of grounded predicates

<sup>1</sup><http://ipc.icaps-conference.org>

such that  $pre(a)$  is a set of grounded predicates representing the precondition of the action  $a$ ,  $eff^-(a)$  is a set of negative effects of the action  $a$ ,  $eff^+(a)$  is a set of positive effects of the action  $a$ , and  $eff^-(a) \cap eff^+(a) = \emptyset$  (in literature, this condition is not always presented, but for a sequential planning we are dealing with, obeying this condition does not bring more expressiveness). The action  $a$  is **applicable** to the state  $s$  if  $pre(a) \subseteq s$ . If the action  $a$  is applicable to the state  $s$ , then the new state  $s'$  obtained after applying the action is  $s' = (s \setminus eff^-(a)) \cup eff^+(a)$ . A **planning operator**  $o$  is a 3-tuple  $(pre(o), eff^-(o), eff^+(o))$  of sets of predicates (not necessarily grounded) and if there exists a substitution  $\Theta$  from variables to constants then  $(pre(o), eff^-(o), eff^+(o))\Theta$  is an action. A **planning domain** is represented by a set of predicates and a set of operators (or actions if the planning domain is grounded). A **planning problem** is represented by a planning domain, a set of constants representing particular objects, an initial state, and a set of goal predicates. A (valid) **plan** is an ordered sequence of actions which leads from the initial state to any goal state containing all the goal predicates. A planning problem is **solvable** if and only if there exists a valid plan. For a deeper insight in this area, see [1].

Planning domains and problems are usually formalized in a LISP-based language called Planning Domain Definition Language (PDDL) [16]. PDDL representation of classical planning problems, we are dealing with in this paper, is often called STRIPS representation.

### III. RELATED WORKS

The idea of using macro-operators (or macro-actions) is not very new, it is advisable to mention REFLECT [17] or Macro Problem Solver [9]. State-of-the-art systems can be divided into two main groups - planner dependent and planner independent. Planner dependent systems MARVIN [18] or SOL-EP version of MACRO-FF [12] are build upon FF planner [3]. Planner independent macro-operator learning systems, on the other hand, can be used with arbitrary planners without changing their source codes. CA-ED version of MACRO-FF [12] (learns macro-operators through analyzing static predicates) or WIZARD [10] (learns macro-operators genetically) are good examples of such planner independent systems. Finally, work described in [13], [14] is also a planner independent macro-operator learning system, but in addition it removes primitive operators replaced by generated macro-operators in all the training plans.

Elimination of unnecessary actions are done by modern planners (like FF [3], LAMA [19] etc.) in terms that actions are eliminated if they are not applicable in any point of the planning process (unreachable actions). Entanglements, introduced in [15], represent connections between operators and initial or goal predicates. By using entanglements we can prune also reachable actions that are (mostly) unnecessary.

### IV. MACRO-OPERATORS

Macro-operators, as mentioned before, act as sequences of primitive operators. Obviously, macro-operators can be under-

stood as ‘shortcuts’ in the search space, because shorter plans need to be explored to find a solution. Basically, the advantage of using macro-operators takes place if the corresponding sequences of instances of primitive operators appear frequently in plans. A macro-operator  $o_{i,j}$  is constructed by assembling operators  $o_i$  and  $o_j$  (in this order) in the following way:

- $pre(o_{i,j}) = pre(o_i) \cup (pre(o_j) \setminus eff^+(o_i))$
- $eff^-(o_{i,j}) = (eff^-(o_i) \cup eff^-(o_j)) \setminus eff^+(o_j)$
- $eff^+(o_{i,j}) = (eff^+(o_i) \cup eff^+(o_j)) \setminus eff^-(o_j)$

By a simple consideration that a macro-operator is a planning operator we can easily extend this construction for larger sequences of primitive operators.

### V. ENTANGLEMENTS

Entanglements [15], [14] refer to connections between planning operators and initial or goal predicates. Such actions (instances of operators) that violate entanglements can be pruned even though these actions might be reachable. It is good for reducing the branching factor. According to [15], we provide formal definitions.

**Definition 1:** Let  $P = \langle \Sigma, s_0, g \rangle$  be a planning problem,  $o$  be a planning operator from  $\Sigma$  and  $p$  be a predicate. The operator  $o$  is *entangled by init (resp. goal)* with the predicate  $p$  in the planning problem  $P$  if and only if  $p \in pre(o)$  (resp.  $p \in eff^+(o)$ ) and there exists a plan  $\pi$  that solves  $P$  and for every action  $a \in \pi$  which is an instance of  $o$  and for every grounded instance  $p_{ground}$  of the predicate  $p$  holds:  $p_{ground} \in pre(a) \Rightarrow p_{ground} \in s_0$  (resp.  $p_{ground} \in eff^+(a) \Rightarrow p_{ground} \in g$ ).

**Definition 2:** Let  $\Sigma$  be a planning domain,  $o$  be a planning operator from  $\Sigma$  and  $p$  be a predicate. The operator  $o$  is *fully entangled by init (resp. goal)* with the predicate  $p$  if and only if there does not exist any solvable planning problem  $P$  over  $\Sigma$  where  $o$  is not entangled by init (resp. goal) with  $p$  in  $P$ . In addition we define a set  $plans(P, o, p, init$  (resp.  $goal)) = \{\pi \mid \pi \text{ is a solution of } P \text{ and satisfy the conditions of entanglement of } o \text{ and } p \text{ regarding def. 1}\}$ .

The entanglement by init (resp. goal) says that in a given planning problem we use only such instances of operators, whose predicates in preconditions (resp. positive effects) correspond with the initial (resp. goal) predicates. The full entanglement extends this for every solvable planning problem in the given domain.

Definition 2 ensures the existence of plans for every solvable planning problem when pruning actions violating the full entanglement conditions. In a general case, each full entanglement somehow restricts these sets of plans allowing only such actions that do not violate the particular full entanglement. However, such restrictions differ regarding the particular full entanglements. The different full entanglements can be used together in such a way that every solvable planning problem remains solvable even if actions violating

all the full entanglements are pruned.

**Definition 3:** Let  $\Sigma$  be a planning domain,  $SFE$  a set of triples  $SFE = \{(o_1, p_1, t_1), \dots, (o_n, p_n, t_n)\}$ , where  $o_i$  is an operator from  $\Sigma$ ,  $p_i$  is a predicate from  $\Sigma$ ,  $t_i \in \{init, goal\}$  and  $o_i$  is fully entangled by  $t_i$  on  $p_i$ . If for every solvable planning problem  $P$  over  $\Sigma \bigcap_{i=1}^n plans(P, o_i, p_i, t_i) \neq \emptyset$  holds, then  $SFE$  is a set of compatible full entanglements.

Detecting (compatible) full entanglements might be done theoretically, but it requires a domain expert who will theoretically prove the (full) entanglements for particular domains. In [15], the (compatible) full entanglements are detected heuristically in terms that it is investigated whether the particular full entanglement holds for all the training plans (modified version incorporates a ‘flaws’ ratio allowing a small number of violations of the entanglement conditions in the training plans).

## VI. COMBINING MACRO-OPERATORS AND ENTANGLEMENTS

As mentioned before in the text, the main issue of using macro-operators rests in the significant increase of the branching factor. On the other hand, eliminating actions via entanglements is a technique that reduces the branching factor. Obviously, we believe that combining the both techniques could result in an improvement of the planning process.

The main question is how macro-operators and entanglements can be combined. In the following lines, we will formally show how to combine macro-operators and (full) entanglements.

*Proposition 1:* Let  $o_{i,j}$  be a macro-operator constructed from planning operators  $o_i$  and  $o_j$  (in this order). Let  $p$  be a predicate. Then, the following statements hold:

- 1) If  $o_i$  is fully entangled by *init* with  $p$ , then  $o_{i,j}$  is also fully entangled by *init* with  $p$ .
- 2) If  $o_j$  is fully entangled by *init* with  $p$ , then  $o_{i,j}$  is also fully entangled by *init* with  $p$  iff  $p \notin eff^+(o_i)$ .
- 3) If  $o_i$  is fully entangled by *goal* with  $p$ , then  $o_{i,j}$  is also fully entangled by *goal* with  $p$  iff  $p \notin eff^-(o_j)$ .
- 4) If  $o_j$  is fully entangled by *goal* with  $p$ , then  $o_{i,j}$  is also fully entangled by *goal* with  $p$ .

*Proof:*

- 1) We know from the definition of entanglements that  $p \in pre(o_i)$ . Considering that  $pre(o_{i,j}) = pre(o_i) \cup (pre(o_j) \setminus eff^+(o_i))$  we get that  $p \in pre(o_{i,j})$ .
- 2) We know from the definition of entanglements that  $p \in pre(o_j)$ . Considering that  $pre(o_{i,j}) = pre(o_i) \cup (pre(o_j) \setminus eff^+(o_i))$  and  $p \notin eff^+(o_i)$  we get that  $p \in pre(o_{i,j})$ .
- 3) We know from the definition of entanglements that  $p \in eff^+(o_i)$ . Considering that  $eff^+(o_{i,j}) = (eff^+(o_i) \cup eff^+(o_j)) \setminus eff^-(o_j)$  and  $p \notin eff^-(o_j)$  we get that  $p \in eff^+(o_{i,j})$ .

- 4) We know from the definition of entanglements that  $p \in eff^+(o_j)$ . Considering that  $eff^+(o_{i,j}) = (eff^+(o_i) \cup eff^+(o_j)) \setminus eff^-(o_j)$  and  $eff^-(o_j) \cap eff^+(o_j) = \emptyset$  we get that  $p \in eff^+(o_{i,j})$ .

We showed that the predicate ( $p$ ) is inherited from the primitive operators ( $o_1$  or  $o_2$ ) if the certain conditions (listed in 1-4) hold. Considering the fact (from the definition of full entanglements) that for every solvable planning problem there exists a (valid) plan such that all instances of  $o_1$  (resp.  $o_2$ ) satisfy the entanglement conditions. It means that the certain instance of the predicate  $p$  (involved in the entanglement) corresponds with some initial or goal predicate. If (consecutive) instances of  $o_1$  and  $o_2$  are assembled into a macro-action (the instance of  $o_{1,2}$ ), then we know that the same instance of  $p$  is also presented in the precondition (resp. positive effects) of this macro-action. Obviously, this instance of  $p$  satisfy the entanglement conditions with respect to  $o_{i,j}$ . ■

We proved that macro-operators assembled from two primitive operators can inherit the entanglements from them. It can be easily extended, because every macro-operator is also a planning operator that can be assembled with other primitive operator or macro-operator, and so on.

## VII. EXAMPLE

Let us consider well known planning domain Blocksworld [20]. In the Blocksworld domain, there is a robotic arm which can move blocks on or off the table or the other blocks. Blocks can be stacked only in tower structures (i.e., no block can be stacked directly on more than one block and at most one block can be stacked on another block). In the Blocksworld domain, we have four planning operators: STACK (stacks a block on another block), UNSTACK (unstacks a block from another block), PICKUP (picks a block from the table), and PUTDOWN (puts a block to the table). Operators STACK and UNSTACK have two arguments (they operate with two blocks), the other operators just one.

From the macro-operators point of view, we can simply observe that the instances of UNSTACK are followed by the instances of PUTDOWN or STACK and the instances of PICKUP are followed by the instances of STACK. It results in a creation of macro-operators UNSTACK-PUTDOWN (unstacks a block from another block and puts it to the table), UNSTACK-STACK (unstacks a block from another block and stacks it to another block) and PICKUP-STACK (picks up a block from the table and stacks it to another block). If no initial or goal state consider situation where the robotic arm holds some block, then we can remove all the primitive operators.

From the entanglements point of view, we can see that to find a solution we have to take down the initial ‘towers’ of blocks and then we have to build the required ‘towers’ of blocks. By a simple observation we can find out that UNSTACK (resp. STACK) operators are (fully) entangled by *init* (resp. *goal*) with a predicate describing which block is stacked on which block. For instance, if a block B is initially

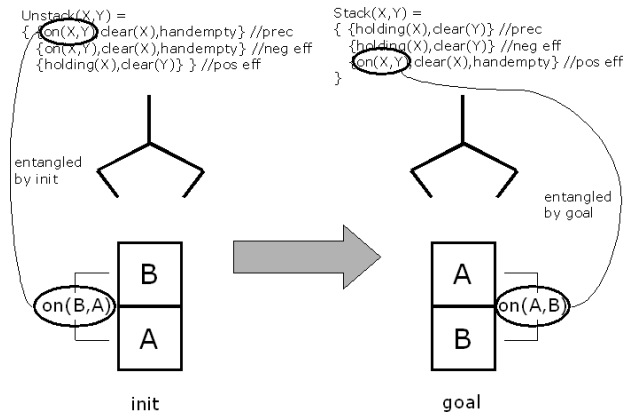


Fig. 1. An example of entanglements in a simple Blocksworld problem

stacked on a block A, then we allow action UNSTACK(B,A), otherwise we prune this action (see fig. 1).

If we put both approaches together, then according to the proposition 1 the (full) entanglements are inherited by the macro-operators from the primitive operators. UNSTACK-PUTDOWN (resp. PICKUP-STACK) has the same arguments as UNSTACK (resp. STACK). It means that inheriting the (full) entanglements results in allowing the same instances, i.e., we allow the same instances of UNSTACK-PUTDOWN as we allowed for UNSTACK (similarly for PICKUP-STACK and STACK). UNSTACK-STACK is the most interesting macro-operator, because it inherits the (full) entanglements from the both UNSTACK and STACK operators. It results in the fact that action UNSTACK-STACK(A,B,C) is allowed if and only if A is initially stacked on B and A is required to be stacked on C.

Let us assume we have a planning problem with  $n$  blocks. It is straightforward that if we use the macro-operators (and remove the primitive operators), then we reduce the depth of the search space to half (all the macro-operators consist of two primitive operators). The question is how many actions (instances of operators or macro-operators) are considered. Here we provide upper estimation for all the cases:

- **Original:**  $2n^2 + 2n$  - UNSTACK and STACK have two arguments, PICKUP and PUTDOWN have only one argument.
- **Macro-operators:**  $n^3 + 2n^2$  - UNSTACK-STACK has three arguments, UNSTACK-PUTDOWN and PICKUP-STACK have two arguments.
- **Entanglements:**  $4n$  - as explained before the (full) entanglements allow to instantiate UNSTACK actions in such a way that we can only unstack a block from its initial position. As we know from the domain description, the block can be stacked on at most one other block. It similarly holds for STACK.
- **Both:**  $3n$  - UNSTACK-PUTDOWN and PICKUP-STACK are affected by the (full) entanglements in the similar way as UNSTACK and STACK. UNSTACK-

STACK is affected in such a way that a block can be moved only from its initial position to its goal position. It results that for every block that is going to move we can have at most one instance of UNSTACK-STACK.

The number of considered actions in particular planning problems do not have to be necessarily in correlation with the branching factor. On the other hand, it may give us an insight how difficult it could be for planners to find a solution. We can see that in the Blocksworld domain combining macro-operators and entanglements results either in the reduction of the depth of the search space and the number of considerable actions. It gives us a positive outlook for using both learning methods (macro-operators and entanglements) together.

## VIII. EXPERIMENTAL EVALUATION

### A. Domains

We chose several planning domains for our experiments to ensure that the proposed approach is generally applicable (rather than specific for a particular planning domain). In particular, we used domains well known from the IPC.

*Depots* is a planning domain from the 3rd IPC. This domain accommodates both blocks and logistics environments. They are combined to form a domain in which trucks can transport crates around and then the crates must be stacked onto pallets at their destinations. The stacking is achieved using hoists, so the stacking problem is like a Blocksworld problem. Trucks can behave like ‘tables’, since the pallets on which crates are stacked are limited.

*Zeno Travel* is a planning domain from the 3rd IPC. This domain involves transporting people around in planes, using different modes of movement: fast and slow.

*Gold Miner* is a planning domain from the 6th IPC learning track. A robot is in a mine and has the goal of reaching a location that contains gold. The mine is organized as a grid with each cell either being hard or soft rock. There is a special location where the robot can either pickup an endless supply of bombs or pickup a laser cannon. The laser cannon can shoot through both hard and soft rock, whereas the bomb can only penetrate soft rock. However, the laser cannon will also destroy the gold if used to uncover the gold location. The bomb will not destroy the gold.

*Rovers* is a planning domain from the 3rd IPC (we slightly updated the operators to satisfy the condition of disjoint sets of positive and negative effects - it does not affect the sequential planning we are dealing with). Inspired by planetary rovers problems, this domain requires that a collection of rovers navigate a planet surface, finding samples and communicating them back to a lander.

*Gripper* is a planning domain from the 1st IPC. In this domain, there is a robot with two grippers. It can carry a ball in each. The goal is to take  $N$  balls from one room to another.

### B. Planners

All the planners we introduce here belong to the most success planners competing on the last IPCs.

Domain	MO added	PO removed	Ent
Depots	2	4	5
Zeno	1	1	2
Gold-Miner	2	3	3
Rovers	1	2	7
Gripper	2	3	4

TABLE I

THE TABLE SHOWS THE RESULTS OF THE LEARNING PHASE (MO = MACRO-OPERATORS ADDED, PO = PRIMITIVE OPERATORS REMOVED AND ENT = ENTANGLEMENTS DETECTED.)

*SATPLAN* [21] (we used version 2006) is a planner that translates the Planning Graph [2] into boolean formulae and then applies a SAT solver that solves it. Straightforwardly, *SATPLAN*'s performance depends mainly on the Planning Graph itself. The Planning Graph, simply said, allows to perform a bunch of actions that do not interfere in one step (it prunes many symmetries). The number of such steps is denoted as Makespan.

*SGPLAN* [22] (we used version 5.22) is a planner that decomposes planning problems in such a way that it satisfies the goals consecutively by a FF-based [3] planner. Such a planner may experience difficulties if a certain goal ordering is essential (for instance in the Blocksworld domain we have to build 'tower' from the ground, not from the middle) or a dead-end (a state from which the goal is unreachable) may be reached during the search.

*LAMA* [19] is a planner that uses multi-heuristic search accommodated with Landmarks [23] (fact that must be true at some point of the planning process), Causal Graph [24] (describes causalities between actions) and FF-based heuristics. *LAMA*'s foundations are from a wider point of view close to *SGPLAN*'s foundations, because Landmark heuristic can provide a certain goal ordering (goal facts are landmarks). On the other hand, *SGPLAN*' goal ordering is implemented in a different way (actually, *SGPLAN*'s documentation do not describe how exactly).

### C. Learning Phase

Regarding the macro-operator generation we took the results gained in [13], [14], where the number of training plans (generated by *SATPLAN* [21] or *SGPLAN* [22]) varied from 3-6. Similarly, regarding the entanglements we took the results gained in [15], [14] or generate the new ones by the method listed also in [15], [14]. The number of training plans (generated by *SATPLAN* [21] only) varied also from 3-6. In the table I we can see the results of the learning phase. The generated macro-operators were assembled in the most cases from a sequence of two primitive operators, except the Gripper domain, where the macro-operators were assembled from a sequence of 5, respectively 6 primitive operators, and the Gold Miner domain, where one of the macro-operators were assembled from a sequence of 4 primitive operators.

The advantage of these methods rests in a little time (tenths of seconds) that was spent on learning. Combining macro-operators and entanglements (according to the proposition 1)

Domain	Macro	Ent	Both
Depots	266%	20%	7%
Zeno	85%	85%	70 %
Gold-Miner	108%	96%	76%
Rovers	59%	100%	59%
Gripper	3152%	51%	1576%

TABLE II

THE TABLE SHOWS HOW MANY ACTIONS IN COMPARISON TO ORIGINAL PROBLEMS (IN AVERAGE) WERE CONSIDERED DURING COMPUTATION (RESULTS OBTAINED BY *LAMA*).

can be also done in a little time.

### D. Comparisons

In the table II we can see the number of actions considered by the planners during searching for plans (obtained by the *LAMA* planner [19]). It is clear (from the definition of entanglements) that the number of actions considered by the planners in the 'entanglement' case cannot be larger than in the original case. However, in the Rovers domain the entanglements were able to prune only unreachable actions, so the number of considered actions remained the same. Regarding the macro-operators, we expected that the number of considered actions will grow (in the Gripper domain it grew rapidly). On the other hand the results showed (thanks to removing replaced primitive operators) that in some cases the number of considered actions is lower. When the both learning techniques are combined it in the most cases (except the Gripper domain) led to the lowest number of considerable actions. As the example (Section VII) indicated the reason of this rests in inheriting (full) entanglements by macro-operators and removing primitive operators replaced by macro-operators. By a simple consideration we can see (in the most cases) that macro-operators define links between primitive operators. It means that (full) entanglements are propagated by these links to operators that were not directly involved in these full entanglements.

In the table III there are presented the results of time comparison<sup>2</sup>. The results were obtained by *SATPLAN*, *SGPLAN* and *LAMA* (see Subsection VIII-B). The cell value '>600' means that the planner did not find the solution within the timeout 600 seconds. The cell value 'err' means that the planner terminated with unexpected error, for instance, *SATPLAN* terminated in the most cases from out of memory reasons.

In the table IV there are presented summarized results for the both time comparison and plan quality. We also provided numbers of problems successfully solved in the time limit of 600 seconds and we pointed out how many of these problems were solved in the littlest time with respect to the particular reformulation (ties are stated in all the corresponding columns). Speed-ups are computed as a geometric mean of  $t_{orig}/t$  ( $t_{orig}$  stands for the running time of the original problem,  $t$  stands for the running time of the reformulated problem). We took into the account only such problems, where

<sup>2</sup>Performed on Core2Duo 2.4GHz, 4GB RAM, Ubuntu Linux

problem	SATPLAN				SGPLAN				LAMA			
	orig	macro	ent	both	orig	macro	ent	both	orig	macro	ent	both
depotprob1817	>600	err	>600	>600	24.47	15.52	0.16	err	331.11	93.68	3.68	0.12
depotprob1916	137.85	err	5.36	0.62	0.40	1.73	80.53	16.90	1.74	2.46	0.14	0.05
depotprob4321	5.25	2.07	0.46	0.02	0.03	0.04	0.00	0.01	4.94	0.25	0.03	0.01
depotprob4398	1.11	2.76	0.29	0.03	0.03	0.13	0.00	0.50	0.23	0.41	0.03	0.01
depotprob4534	>600	err	>600	218.62	>600	0.53	>600	0.02	362.81	1.39	5.86	0.02
depotprob5646	0.38	6.08	0.09	0.14	0.02	0.05	0.00	0.01	0.17	0.25	0.02	0.00
depotprob5656	222.28	143.33	5.92	1.10	410.61	0.32	0.24	0.03	>600	0.53	3.41	0.02
depotprob6178	6.92	26.11	1.49	0.19	0.10	0.27	0.02	371.19	11.61	1.44	0.06	0.02
depotprob6587	3.36	19.02	0.59	0.08	0.07	0.41	0.01	err	0.42	0.98	0.05	0.02
depotprob7615	>600	err	>600	>600	8.06	1.88	>600	err	>600	5.71	>600	0.06
depotprob7654	10.08	16.45	1.41	0.11	0.09	0.07	0.00	28.40	1.45	0.59	12.32	0.01
depotprob8715	36.04	err	8.70	0.96	0.27	2.84	0.04	0.05	1.65	6.35	0.17	0.04
depotprob9876	>600	err	>600	67.86	0.23	0.42	516.15	19.84	580.18	1.27	0.19	0.02
ztravel-3-7	1.87	2.62	1.54	2.36	0.01	0.00	0.00	0.01	0.04	0.05	0.04	0.04
ztravel-3-8a	1.68	2.14	1.33	1.75	0.01	0.01	0.00	0.00	0.05	0.04	0.04	0.04
ztravel-3-8b	0.86	1.01	0.62	0.86	0.01	0.00	0.00	0.00	0.04	0.03	0.04	0.02
ztravel-3-10	3.76	4.17	3.71	5.04	0.01	0.02	0.00	0.02	0.05	0.06	0.05	0.06
ztravel-5-10	34.23	48.19	22.23	33.91	0.22	0.24	0.19	0.20	0.25	0.22	0.20	0.18
ztravel-5-15a	92.57	30.93	40.84	17.17	0.12	0.06	0.10	0.04	0.48	0.19	0.37	0.12
ztravel-5-15b	err	err	err	err	0.56	0.59	0.46	0.51	0.63	0.50	0.49	0.40
ztravel-5-20a	err	err	err	err	0.87	0.75	0.67	0.55	1.22	0.87	1.20	0.52
ztravel-5-20b	err	err	err	err	1.07	0.77	0.87	0.52	1.57	0.75	1.15	0.54
ztravel-5-25a	err	err	err	err	1.72	1.05	1.35	0.70	2.82	0.98	2.90	0.56
ztravel-5-25b	err	err	err	err	0.56	0.58	0.44	0.35	7.22	1.42	3.78	0.88
gold-miner-7x7-01	5.98	6.06	4.90	3.32	err	0.02	0.00	0.01	0.30	0.04	0.05	0.02
gold-miner-7x7-02	4.39	4.26	3.60	2.43	err	0.01	0.00	0.01	0.16	0.04	0.03	0.02
gold-miner-7x7-03	4.12	4.01	3.44	2.36	err	0.01	0.00	0.00	0.29	0.04	0.04	0.02
gold-miner-7x7-04	9.31	10.51	7.58	5.52	err	0.01	0.00	0.00	0.21	0.04	0.02	0.02
gold-miner-7x7-05	9.71	9.99	8.17	5.62	err	0.01	0.01	0.00	0.38	0.04	0.03	0.02
gold-miner-7x7-06	6.05	6.34	5.02	3.44	err	0.01	0.00	0.01	0.18	0.04	0.02	0.02
gold-miner-7x7-07	6.14	5.82	5.05	3.26	err	0.02	0.00	0.01	0.04	0.04	0.02	0.02
gold-miner-7x7-08	3.12	2.81	2.61	1.68	err	0.01	0.00	0.01	>600	0.03	0.02	0.02
gold-miner-7x7-09	4.26	4.26	3.63	2.45	err	0.01	0.00	0.00	0.14	0.04	0.03	0.02
gold-miner-7x7-10	5.93	6.05	4.92	3.39	err	0.01	0.01	0.01	0.29	0.04	0.05	0.01
rovers1425	0.42	0.33	0.37	0.28	0.08	0.01	0.00	0.00	0.02	0.02	0.02	0.01
rovers4135	1.18	0.76	0.95	0.53	0.87	0.02	0.01	0.01	0.03	0.03	0.04	0.03
rovers4621	2.29	2.12	1.71	1.52	2.07	0.03	0.03	0.03	0.06	0.08	0.07	0.06
rovers5142	0.75	0.39	0.57	0.25	0.09	0.01	0.01	0.01	0.04	0.02	0.04	0.02
rovers5146	0.08	0.04	0.06	0.03	0.02	0.00	0.00	0.00	0.01	0.01	0.01	0.01
rovers5624	4.24	2.92	3.05	1.65	0.09	0.01	0.01	0.01	0.06	0.05	0.06	0.04
rovers6152	0.63	0.66	0.50	0.45	0.03	0.01	0.02	0.01	0.04	0.03	0.03	0.02
rovers7126	0.40	0.36	0.31	0.24	0.02	0.01	0.00	0.00	0.02	0.02	0.02	0.02
rovers7182	48.07	41.45	31.31	24.60	3.72	0.11	0.11	0.08	0.21	0.18	0.21	0.18
rovers8271	0.12	0.07	0.11	0.05	0.06	0.00	0.00	0.00	0.02	0.00	0.02	0.02
rovers8327	15.71	11.47	10.57	7.50	3.14	0.06	0.09	0.04	0.17	0.12	0.16	0.11
gripper11	>600	5.71	>600	2.90	0.03	0.22	0.00	0.02	0.02	0.75	0.02	0.40
gripper12	>600	7.91	>600	3.88	0.04	0.29	0.00	0.07	0.04	0.93	0.02	0.50
gripper13	>600	10.68	>600	5.36	0.04	0.34	0.00	0.05	0.02	1.14	0.02	0.62
gripper14	>600	14.98	>600	6.96	0.04	0.44	0.00	0.05	0.04	1.40	0.03	0.74
gripper15	>600	20.59	>600	9.38	0.05	0.54	0.01	0.05	0.04	1.65	0.03	0.88
gripper16	>600	26.03	>600	11.89	0.05	0.64	0.00	0.06	0.05	1.98	0.04	1.05
gripper17	>600	err	>600	15.46	0.05	0.77	0.00	0.17	0.05	2.30	0.04	1.23
gripper18	>600	err	>600	20.08	0.06	0.90	0.02	0.10	0.06	2.74	0.04	1.42
gripper19	>600	err	>600	24.64	0.06	1.06	0.01	0.11	0.07	3.12	0.05	1.66
gripper20	>600	err	>600	30.81	0.07	1.24	0.01	0.13	0.07	3.59	0.06	1.89

TABLE III

THE TABLE SHOWS COMPARISON OF RUNNING TIMES (IN SECONDS - TIMEOUT WAS SET TO 600 SECONDS) OF ORIGINAL PROBLEMS AND PROBLEMS REFORMULATED BY MACRO-OPERATORS, ENTANGLEMENTS AND THE BOTH.

Domain	Planner	# Solved (# Best)				Speedup			Quality		
		Orig	Mcr	Ent	Both	Mcr	Ent	Both	Mcr	Ent	Both
Depots	SATPLAN	9 (0)	7 (0)	9 (0)	11 (11)	0.4	8.0	56.9	1.08	1.07	1.12
Depots	SGPLAN	12 (2)	13 (1)	11 (8)	10 (2)	1.0	5.7	0.3	0.87	1.19	1.13
Depots	LAMA	11 (0)	13 (0)	12 (0)	13 (13)	4.1	26.3	340.4	1.04	1.12	1.09
Zeno	SATPLAN	6 (0)	6 (0)	6 (5)	6 (1)	1.0	1.4	1.2	1.02	1.00	0.94
Zeno	SGPLAN	11 (0)	11 (2)	11 (6)	11 (7)	1.6	2.6	2.0	0.88	1.00	0.87
Zeno	LAMA	11 (2)	11 (1)	11 (3)	11 (10)	1.6	1.2	2.2	0.92	0.99	0.89
Gold-Miner	SATPLAN	10 (0)	10 (0)	10 (0)	10 (10)	1.0	1.2	1.8	1.01	1.01	1.06
Gold-Miner	SGPLAN	0 (0)	10 (1)	10 (9)	10 (5)	N/A	N/A	N/A	N/A	N/A	N/A
Gold-Miner	LAMA	9 (0)	10 (0)	10 (4)	10 (10)	4.8	6.3	10.3	4.67	4.59	4.67
Rovers	SATPLAN	11 (0)	11 (0)	11 (0)	11 (11)	1.4	1.3	2.0	1.07	1.04	1.05
Rovers	SGPLAN	11 (0)	11 (6)	11 (8)	11 (11)	16.5	24.2	28.5	1.08	1.07	1.08
Rovers	LAMA	11 (4)	11 (6)	11 (2)	11 (10)	1.5	1.0	1.3	0.99	1.00	1.00
Gripper	SATPLAN	0 (0)	6 (0)	0 (0)	10 (10)	N/A	N/A	N/A	N/A	N/A	N/A
Gripper	SGPLAN	10 (0)	10 (0)	10 (10)	10 (0)	0.1	17.7	0.7	1.34	1.25	1.34
Gripper	LAMA	10 (2)	10 (0)	10 (10)	10 (0)	0.0	1.3	0.0	1.00	1.00	1.00

TABLE IV

THE TABLE SHOWS THE RESULTS. PROBLEMS ARE MARKED AS SOLVED IF SUCCESSFULLY SOLVED WITHIN 10 MINUTES. BEST DENOTES THE NUMBER OF PROBLEMS SOLVED IN THE LITTLEST TIME (TIES ARE STATED IN ALL THE CORRESPONDING COLUMNS). SPEED-UPS ARE COMPUTED AS A GEOMETRIC MEAN OF  $t_{orig}/t$ , PLANS QUALITY IS COMPUTED AS A GEOMETRIC MEAN OF  $|\pi_{orig}|/|\pi|$ .

the original and reformulated problem were both successfully solved. Similarly, quality of plans is computed as a geometric mean of  $|\pi_{orig}|/|\pi|$ . Speed-ups as presented here give us an overview of the potential improvement of the planning process achieved by the learning techniques. However, in cases when the original problems were unsolved or, on the other hand, solved in a very little time, then the speed-up values may become inaccurate (to get deeper insight about speed-ups and time comparisons at all, remember the table III).

### E. Discussion

The presented results showed that combining macro-operators and entanglements is reasonable and can help to improve the planning process.

1) *SATPLAN*: By using entanglements we can prune unnecessary actions. It makes the Planning Graph less complex even though makespan mostly remained the same. The experiments showed speed-ups in all the tested problems that were solved within the timeout. The experiments showed also speed-ups in the Rovers domain (entanglements) even though there have not been pruned any reachable actions - entanglements probably eased the computation of the Planning Graph. Potential success of using macro-operators rests in the reduction of makespan. For instance in the Gripper domain, SATPLAN was able to solve problems with macro-operators in two steps (makespan=2) even though the original problems were unsolved. However, if makespan grows or is reduced only slightly, it may result in slow-down, because the Planning Graph can be much more complex (increased branching factor). It happened in many tested problems (except the Gripper domain). Combining the both approaches (macro-operators and entanglements) brought benefits like reducing makespan and making the Planning Graph less complex. The experiments showed significant improvements in the Depots and Gripper domains. Only in the Zeno travel domain the results were slightly worse than in the entanglement version.

2) *SGPLAN*: In the Gold Miner domain, the SGPLAN's behavior was weird, because in all the tested original problems it unexpectedly terminated after about 3 minutes of running. Using macro-operators resulted in increasing of SGPLAN's performance, except The Gripper domain, where the macro-operators were more complex and the number of their instances was very high. On the other hand, using macro-operators in the Gripper domain resulted in a significant increase of quality of plans. Entanglements brought an improvement as we expected, however, in a few Depots problems the performance rapidly decreased. It was caused by the fact that pruning actions via entanglements in the Depots domain makes dead-ends. As we mentioned dead-ends together with certain goal ordering make trouble to SGPLAN. On the other hand the quality of plans in the 'entangled' Depots domain significantly increased. Combining macro-operators and entanglements, however, did not perform very well, for instance in the Depots domain the results were significantly worse. Promising results were gained only in the Rovers domain.

3) *LAMA*: The success of learning methods stands on reducing the number of explored nodes or simplifying the computation of the heuristics. Macro-operators helped LAMA to reduce the depth of explored state space which resulted in performance improvement, except in the Gripper domain, where slow-down was caused by the extreme number of macro-operators' instances. Entanglements, on the other hand, helped LAMA to simplify the computation of heuristics which also resulted in the performance improvement in almost all the tested problems (depotprob7654 is only one exception). Combining macro-operators and entanglements brought additional improvement of the planning process (except the Gripper domain, where the extreme number of macro-operators' instances were not reduced much by the entanglements). It is the worth of a mention that in the Depots domain all the problems were solved within the hundredths of seconds (at most 0.12s) ! In the Gold Miner domain the original problems were solved in more than four times worse plan quality - probably caused by

LAMA's heuristics.

4) *Concluding Remarks*: By summarizing the results we obviously found out that the success of the learning methods rests in the planner we are using. On the other hand, in the most cases the results were better in reformulated problems (macro-operators, entanglements, the both) than in the original ones. It showed us the reasonability of such approaches. A promising point of using the combined approach rests in the fact that in the most of the tested cases the harder problem we reformulated bigger speed-up we obtained. However, we should raise a question how the learning methods can be further improved. We believe that there is still a space for future investigation, especially in terms, how learned knowledge can be efficiently evaluated. By this we mean to decide in a reasonable time, for instance, whether a macro-operator should be generated or not. Second question we should raise is about theoretical aspects of the learning techniques. Obviously, when we eliminating actions or primitive operators, then it can break the completeness of the planning process. The IPC benchmarks we used here for the evaluation differ only by the number of objects. It resulted in the fact that knowledge learnt from the several simple training plans is valid also for more complex problems (in the particular domain). However, in the real world applications it might cause troubles, because initial or goal predicates might be very different there.

## IX. CONCLUSIONS

Learning for planning seems to be a good direction for improving the planning process. The presented results showed that combining different learning techniques together, in this case macro-operators and eliminating actions (via entanglements), can bring a promising outcome.

Obviously, there remains a space for improvements. For instance, it is possible to adapt the methods for macro-operator generation to be able to consider entanglements during the process where the primitive operators are selected for assemblage into macro-operators. We should investigate more thoroughly how the learned knowledge can be efficiently evaluated, because it is quite necessary to decide whether learned knowledge should be kept or thrown away. The remaining question should discuss whether the evaluation of learned knowledge should be planner dependent or independent. The experiments we made here pointed out that the performance of the planners might be very different when certain knowledge is used. It might bring doubts about planner independent evaluation of learned knowledge. On the other hand it should be studied more thoroughly to make brighter conclusions.

Removing primitive operators or some actions from domains might break the completeness of planning process (some originally solvable problems might become unsolvable). It may happen very occasionally for problems like IPC benchmarks (in experiments we made here it never happened), because they differ by the number of objects, not the kinds of initial states or goals. Studying the theoretical aspects of this might help us to identify, for instance, which problems become unsolvable when some reformulation is applied etc. It will

be necessary for the real world applications. Additionally, handling with time or consumable resources is essential for the real world applications. It will be necessary to extend the learning techniques also for temporal planning and planning with consumable resources.

## ACKNOWLEDGMENTS

The research is supported by the Czech Science Foundation under the project no. 201/08/0509.

## REFERENCES

- [1] M. Ghallab, D. Nau, and P. Traverso, *Automated planning, theory and practice*. Morgan Kaufmann Publishers, 2004.
- [2] A. Blum and M. Furst, "Fast planning through planning graph analysis," *Artificial Intelligence*, vol. 90, no. 1-2, pp. 281-300, 1997.
- [3] J. Hoffmann and B. Nebel, "The FF planning system: Fast plan generation through heuristic search," *Journal of Artificial Intelligence Research (JAIR)*, vol. 14, pp. 253-302, 2001.
- [4] M. Helmert, "The fast downward planning system," *Journal of Artificial Intelligence Research (JAIR)*, vol. 26, pp. 191-246, 2006.
- [5] B. Bonet and H. Geffner, "Planning as heuristic search: New results," in *Proceedings of ECP*, 1999, pp. 360-372.
- [6] H. Kautz and B. Selman, "Planning as satisfiability," in *Proceedings of ECAI*, 1992, pp. 359-363.
- [7] M. B. Do and S. Kambhampati, "Planning as constraint satisfaction: Solving the planning graph by compiling it into csp," *Artificial Intelligence*, vol. 132, pp. 151-182, 2001.
- [8] K. Erol, D. S. Nau, and V. S. Subrahmanian, "Complexity, decidability and undecidability results for domain-independent planning," *Artificial Intelligence*, vol. 76, pp. 75-88, 1995.
- [9] R. Korf, "Macro-operators: A weak method for learning," *Artificial Intelligence*, vol. 26, no. 1, pp. 35-77, 1985.
- [10] M. Newton, J. Levine, M. Fox, and D. Long, "Learning macro-actions for arbitrary planners and domains," in *Proceedings of ICAPS*, 2007, pp. 256-263.
- [11] A. Coles, M. Fox, and A. Smith, "Online identification of useful macro-actions for planning," in *Proceedings of ICAPS*, 2007, pp. 97-104.
- [12] A. Botea, M. Enzenberger, M. Müller, and J. Schaeffer, "Macro-ff: Improving ai planning with automatically learned macro-operators," *Journal of Artificial Intelligence Research*, vol. 24, pp. 581-621, 2005.
- [13] L. Chrpá, "Generation of macro-operators via investigation of action dependencies in plans," *Knowledge Engineering Review*, 2010, to appear.
- [14] L. Chrpá, "Learning for classical planning," Ph.D. dissertation, Charles University in Prague, 2009.
- [15] L. Chrpá and R. Barták, "Reformulating planning problems by eliminating unpromising actions," in *Proceedings of SARA*. AAAI press, 2009, pp. 50-57.
- [16] M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "Pddl the planning domain definition language," Tech. Rep., 1998.
- [17] C. Dawson and L. Siklóssy, "The role of preprocessing in problem solving systems," in *Proceedings of IJCAI*, 1977, pp. 465-471.
- [18] A. Coles and K. A. Smith, "Marvin: A heuristic search planner with online macro-action learning," *Journal of Artificial Intelligence Research (JAIR)*, vol. 28, pp. 119-156, 2007.
- [19] S. Richter and M. Westphal, "The lama planner using landmark counting in heuristic search," in *Proceedings of IPC*, 2008.
- [20] J. Slaney and S. Thiébaux, "Blocks world revisited," *Artificial Intelligence*, vol. 125, no. 1-2, pp. 119-153, 2001.
- [21] H. Kautz, B. Selman, and J. Hoffmann, "Satplan: Planning as satisfiability," in *Proceedings of IPC*, 2006.
- [22] C.-W. Hsu and B. W. Wah, "The sgplan planning system in ipc-6," in *Proceedings of IPC*, 2008.
- [23] J. Hoffmann, J. Porteous, and L. Sebastia, "Ordered landmarks in planning," *Journal of Artificial Intelligence Research (JAIR)*, vol. 22, pp. 215-278, 2004.
- [24] C. Knoblock, "Automatically generated abstractions for planning," *Artificial Intelligence*, vol. 68, no. 2, pp. 243-302, 1994.