An Automatic Algorithm Selection Approach for Planning

Mauro Vallati, Lukáš Chrpa, Diane Kitchin
School of Computing and Engineering
University of Huddersfield
Email: {m.vallati, l.chrpa, d.kitchin}@hud.ac.uk

***Abstract*—Despite the advances made in the last decade in automated planning, no planner outperforms all the others in every known benchmark domain. This observation motivates the idea of selecting different planning algorithms for different domains. Moreover, the planners' performances are affected by the structure of the search space, which depends on the encoding of the considered domain. In many domains, the performance of a planner can be improved by exploiting additional knowledge, extracted in the form of macro-operators or entanglements.**

In this paper we propose ASAP, an automatic Algorithm Selection Approach for Planning that: (i) for a given domain initially learns additional knowledge, in the form of macro-operators and entanglements, which is used for creating different encodings of the given planning domain and problems, and (ii) explores the 2 dimensional space of available algorithms, defined as encodings–planners couples, and then (iii) selects the most promising algorithm for optimising either the runtimes or the quality of the solution plans.

I. Introduction

Although in the last decade the performance of domain-independent planners has significantly improved, there is no planner that outperforms all others in every benchmark domain. The performance of current planning systems is typically affected by the structure of the search space, which depends on the planning domain and its considered encoding. In many domains, the planning performance can be improved by deriving and exploiting knowledge about the domain and problem structure that is not explicitly given in the input formalization, and that can be used for optimizing the planner behavior.

These observations motivate the idea of extracting additional knowledge about the planning domains and automatically selecting the most promising planning algorithm, exploiting such knowledge, for a given domain.

In this paper we propose ASAP, an automatic algorithm selection approach for planning that: (i) for a given domain initially learns additional knowledge, in the form of macro-operators and entanglements (inner and outer), which is used for creating different encodings of the given planning domain and problems (i.e. planning domain/problem reformulation), and (ii) explores the 2 dimensional space encodings (e)–planners (p), and then (iii) selects the best algorithm $\langle e, p\rangle$ for optimising the runtimes (ASAPs) or the quality of the solution plans (ASAPq).

In the proposed approach, each *algorithm* has two dimensions: one dimension is represented by different encodings of a given domain, the other is represented by existing high-performance domain-independent planners. We decided to consider each couple $\langle e, p\rangle$ as a different algorithm because the different knowledge carried in the generated encodings, e, makes even the same planner p perform very differently.

We are not aware of other completely automated planning systems exploiting a *pure* algorithm selection approach, in the sense that they automatically select a single algorithm for solving a specific class of planning problems. If we include the portfolio-based approach for planning, which can be considered as a superset of the algorithm selection one, our approach is related to the work of Roberts and Howe [13], [19], PbP2 [9], [10] and Fast Downward Stone Soup [21], with some significant differences.

The major difference between all the approaches above and ASAP is that we made a domain-specific selection of a single algorithm, which is defined by a couple encoding–planner. Moreover, the Roberts and Howe approaches select the planners to exploit online, while we select the algorithm offline. Additionally the knowledge generated by the Roberts and Howe systems is domain-independent, while the knowledge generated and exploited by ASAP is domain-specific.

PbP2 learns a domain-specific portfolio. It incorporates seven planners it can choose from. It lets them learn macro-actions for the given domain, and runs up to three best-performing ones in a round-robin fashion with learned time slots. What differentiates our approach from PbP2, is that (i) we generate new encodings of given domains by looking for both macro-operators and entanglements, (ii) we explore the two-dimensional algorithm space encodings–planners, and (iii) we select only one algorithm to exploit on a domain.

Fast Downward Stone Soup is a recent approach to selecting and combining a set of forward-state planning techniques included in the well known domain-independent planner Fast Downward [11]. Their approach is domain-independent, it does not extract any additional knowledge from the planning domains (in the form of macro-operators or entanglements). It exploits a statical combination of several different planning techniques for solving a single problem.

In the rest of the paper, first we give the necessary background on classical planning and problem reformulations, then we describe the ASAP approach, we present and discuss

the experimental results and finally we give conclusions.

II. Classical Planning

Classical planning deals with finding a (partially or totally ordered) sequence of actions transforming the static, deterministic and fully observable environment from some initial state to a desired goal state. In the classical representation *atoms* are predicates. *States* are defined as sets of ground predicates. A *planning operator* $o = (name(o), pre(o), eff^-(o), eff^+(o))$ is specified such that $name(o) = op\_name(x_1, \ldots, x_k)$ (*op_name* is an unique operator name and $x_1, \ldots x_k$ are variable symbols (arguments) appearing in the operator), $pre(o)$ is a set of predicates representing operator's precondition, $eff^-(o)$ and $eff^+(o)$ are sets of predicates representing operator's negative and positive effects. *Actions* are ground instances of planning operators. An action $a = (pre(a), eff^-(a), eff^+(a))$ is *applicable* in a state s if and only if $pre(a) \subseteq s$. Application of a in s (if possible) results in a state $(s \setminus eff^-(a)) \cup eff^+(a)$.

A *planning domain* is specified via sets of predicates and planning operators. A *planning problem* is specified via a planning domain, initial state and set of goal atoms. A *solution plan* is a sequence of actions such that a consecutive application of the actions in the plan (starting in the initial state) results in a state that satisfies the goal.

III. Planning Problem Reformulations

Analogously to the possibility that a planning system can be implemented in many different ways, so planning domains and problems can be also encoded in several different ways. Typically, environment and action descriptions correspond with real situations which produces useful outputs for agents (or robots) that they can easily execute. On the other hand, sometimes such an encoding is not very efficient and therefore some additional planner independent knowledge (e.g. macro-operators) is often included to increase the efficiency of planning engines.

As a running example we use the well known BlocksWorld domain. It consists of four operators: pickup(?x) refers to a situation when a robotic hand picks-up a block ?x from the table, putdown(?x) refers to a situation when a robotic hand puts-down the block ?x it is holding to the table, unstack(?x,?y) refers to a situation when a robotic hand unstacks a block ?x from ?y, and stack(?x,?y) refers to a situation when a robotic hand stacks a block ?x to ?y.

A. Macro-operators

A macro-operator encapsulates a sequence of (primitive) planning operators and can be represented as an ordinary planning operator. In Blocksworld, it may be observed that instances of the operator unstack(?x ?y) are followed by instances of the operator putdown(?x). Hence, it is reasonable to assemble these operators into a macro-operator unstack-putdown(?x ?y). Creating macro-operators, which can be understood as 'short-cuts' in the state space, is therefore a well known and studied approach which in some cases can speed up plan generation considerably [1], [16]. Macro-operators can be added into planning domains and reformulated domains can be passed to any planning engine. To raise the efficiency of the planning process, an approach [5] besides generating new macro-operators also removes some primitive operators which are very likely useless. In our example, when the new macro-operator unstack-putdown(?x ?y) is created, then it may be observed that the primitive operator putdown(?x) is useless (unless an initial state consists of a situation where the robotic hand holds some block).

B. Entanglements

Entanglements [3] are relations between planning operators and atoms (predicates). Entanglements aim to capture the causal relationships characteristic for a given class of planning problems which in many cases enable a reduction of the branching factor in the state space. There are two kinds of entanglements, outer and inner entanglements.

Outer entanglements [4] are relations between planning operators and initial or goal atoms (predicates) which refers to situations where to solve a given planning problem we need only such instances of operators where instances of a certain predicate in an operator's precondition or positive (add) effects respectively are present in the initial state or goal situation respectively. In BlocksWorld, it can be observed that unstacking blocks only occurs from their initial positions. In this case an 'entanglement by init' will capture that if an atom on(a b) is to be achieved for a corresponding instance of operator unstack(?x ?y) (unstack(a b)), then the atom is an initial atom. Similarly, it may be observed that stacking blocks only occurs to their goal positions. Then, an 'entanglement by goal' will capture that atom on(b a) achieved by a corresponding instance of operator stack(?x ?y) (stack(b a)) is a goal atom. Outer entanglements can be easily encoded into planning domain models, i.e., the original domain model is reformulated (for details, see [4]).

Inner entanglements [3] are relations between pairs of planning operators and predicates which refer to situations where one operator is an exclusive 'achiever' or 'consumer' of a predicate to or from another operator. In the Blocksworld it may be observed that operator pickup(?x) achieves predicate holding(?x) exclusively for operator stack(?x,?y) (and not for operator putdown(?x)), i.e., pickup(?x) is 'entangled by succeeding' stack(?x,?y) with holding(?x). Similarly, it may be observed that predicate holding(?x) for operator putdown(?x) is exclusively achieved by operator unstack(?x ?y) (and not by operator pickup(?x)), i.e., putdown(?x) is 'entangled by preceding' unstack(?x ?y) with holding(?x). Inner entanglements can

Domain	ASAPs	ASAPq	Total
Blocksworld	⟨Both, FF⟩	⟨Both, FF⟩	35
Depots	⟨Outer, FF⟩	⟨Both, LPG⟩	35
Gripper	⟨Outer, SGPlan⟩	⟨Outer, Mp⟩	14
Gold Miner	⟨Macro, FF⟩	⟨Both, SatPlan⟩	35
Matching-BW	⟨Macro, LPG⟩	⟨Outer, LPG⟩	35
Parking	⟨Original, FF⟩	⟨Inner, Lama⟩	21
Rovers	⟨Macro, LPG⟩	⟨Macro, Lama⟩	21
Satellite	⟨Outer, LPG⟩	⟨Outer, LPG⟩	21
TPP	⟨Both, LPG⟩	⟨Outer, Lama⟩	35

Table I
FOR EVERY DOMAIN, THE COUPLE SELECTED BY ASAPs AND ASAPq, AND THE TOTAL NUMBER OF AVAILABLE ALGORITHMS.

be also encoded into planning domain models (for details, see [3]).

IV. The Proposed Approach

ASAP includes the following existing high performance domain-independent planners: Lama-11 [17], LPG [8], Metric-FF [12], Mp [18], Probe [15], SatPlan [14] and SGPlan [2]. We selected them due to their good performances in International Planning Competitions (IPC) and the different techniques that they exploit.

The learning phase of ASAP is composed of four steps: (i) extraction of macro-operators and removal of useless primitive operators, (ii) detection of entanglements and encoding them into new planning domains/problems, (iii) generation of all the algorithms as couples $\langle e, p\rangle$, (iv) measurement of the performances of the available algorithms, and (v) selection of the most promising algorithm for solving the testing instances.

Macro-operators and entanglements are extracted using the approach described, respectively, in [5] and [3] on plans generated by Metric-FF, exploiting the original domain encodings, on training problems. Through these techniques ASAP is able to generate at most four new encodings per domain: Macros, which includes macro-operators and excludes some original operators; Inner, which includes inner entanglements; Outer, which includes outer entanglements; Both, which considers both inner and outer entanglements. The maximum number of algorithms per domain is 35, which arises from 7 included planners that can be used with 5 different encodings.

The current version of ASAP runs the available algorithms on training problems. The performances are measured in terms of CPU time required for solving each training instance, number of actions of the solutions found, and the number of solved problems. The performances of each algorithm $\langle e, p\rangle$ are then compared in order to select the most promising one to execute on testing problems. ASAP has two different versions: ASAPs which selects the algorithms for optimising runtimes, and ASAPq which optimises the quality of the plans (in classical STRIPS planning the quality is measured by plan length, i.e., shorter better).

For selecting the most promising algorithm in terms of runtime, ASAPs uses the time IPC score. It is a value, firstly introduced in IPC-6 [7], which considers runtimes and number of solved problems together. It is very useful because it synthesizes different aspects of planners' performance in a single value, that can then be compared through different planners. ASAPs selects the couple which achieved the best IPC score on the learning problems; if more algorithms achieved the same score some secondary criteria are used. These criteria include the number of solved problems, the number of problems in which the couple has been the fastest and the mean CPU time on solved problems.

The method used by ASAPq is similar, but it is considering the quality of plans (in terms of number of actions) instead of the CPU times. For the incremental planners[1], i.e. LPG and Lama, the best solution found within the CPU time limit is considered.

The time and quality IPC score are determined as defined for IPC-7 [6]. The time score of an algorithm $\mathcal{A}$ for a planning problem P is defined as *Score*$(\mathcal{A}, P)$, which is 0 if P is unsolved, and $1/(1+\log_{10}(T_P(\mathcal{A})/T_P^*))$ otherwise, where T_P^* is the lowest measured CPU time to solve problem P and $T_P(\mathcal{A})$ denotes the CPU time required by $\mathcal{A}$ to solve problem P. Higher values of the speed score indicate better performance. The quality score is defined as *Score*$(\mathcal{A}, p)$, which is 0 if p is unsolved, and $Q_p^*/Q(\mathcal{A}_p)$ otherwise ($Q_p^* \leq Q(\mathcal{A})_p$ for any $\mathcal{A}$). Quality is measured in terms of number of actions. The IPC score on a set of problems is given by the sum of the scores achieved on each considered problem.

V. Experimental Analysis

In this section, we present the results of a preliminary experimental study examining the *effectiveness* of the knowledge generated and exploited by ASAP under the form of selected algorithms $\langle e, p\rangle$.

A. Experimental Setup

We considered problem instances from 9 well-known benchmark domains used in the learning tracks of International Planning Competitions: Blocksworld (IPC-7), Depots (IPC-7), Gripper (IPC-7), Gold-miner (IPC-6), Matching-BW (IPC-6), Parking (IPC-6/7), Rovers (IPC-7), Satellite (IPC-7) and TPP (IPC-7). These domains were selected because they are suitable for reformulations. Some domains used in learning tracks are not suitable for extracting additional knowledge in the form of macros or entanglements since they have a very small number of operators (1 or 2). In other domains, since Metric-FF was not able to solve any training problem, ASAP was not able to derive any type of knowledge. In such domains, the comparison would be only between basic solvers, which is not the focus of this paper.

[1] An incremental planner produces a sequence of solutions with increasing plan quality which are generated with increasing CPU times.

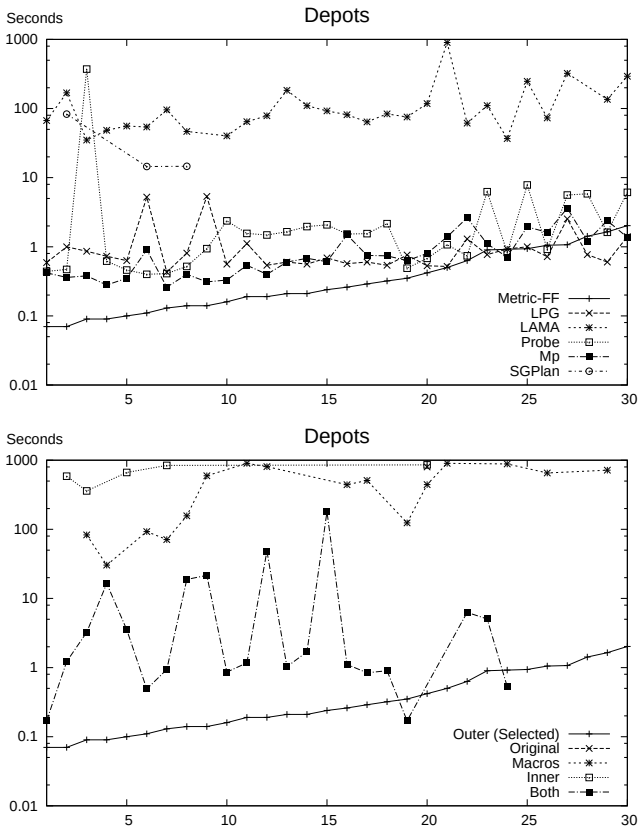

Figure 1. CPU time (log. scale) of the selected couple w.r.t. the couples exploiting the same encoding (upper plot) and couples exploiting the same planner (lower plot) on benchmark problems of Depots domain.

For each domain, we used the existing 30 benchmark problems as testing instances. As training problems we used 30 training problems from those provided by organizers, whenever available; otherwise we generated circa 30 instances (easier than the ones used for testing the approach) through available random generators.

A runtime cutoff of 900 CPU seconds (15 minutes, as in the learning tracks of IPC) was used for both learning and testing runs. All the experiments were run on 3.0 Ghz machine CPU with 2GB of RAM.

B. Results on Selected Domains

Table I shows, for every domain, the algorithm selected by ASAPs and ASAPq, and the total number of available algorithms. It is interesting to note that the selected domains encodings changes frequently through benchmark domains. On the other hand, the planners most frequently included in the couples selected by ASAPs are Metric-FF and LPG, only in Gripper SGPlan is included. On the contrary, for optimising the quality, almost all the different planners have been selected. The total number of available algorithms can be smaller than 35 in the case that macro-operators or some type of entanglements were not found.

Domain	BestS	Time IPC		% Solved	
		ASAPs	BestS	ASAPs	BestS
Bw	Probe	**30.0**	5.8	**100.0**	66.7
Depots	Probe	**30.0**	8.7	100.0	100.0
Gripper	LPG	**30.0**	5.0	**100.0**	33.3
Gold-m	Mp	**30.0**	16.5	100.0	100.0
M-BW	Lama	**30.0**	8.7	**100.0**	80.0
Parking	FF	5.0	5.0	16.7	16.7
Rovers	LPG	**28.0**	22.0	93.3	93.3
Satellite	LPG	**29.0**	27.8	96.7	**100.0**
TPP	Lama	**20.0**	3.4	**66.7**	33.3
All above		**232.0**	102.9	**85.9**	69.3

Domain	BestQ	Quality IPC		% Solved	
		ASAPq	BestQ	ASAPq	BestQ
Bw	Probe	**29.5**	18.1	**100.0**	66.7
Depots	Probe	**30.0**	26.2	100.0	100.0
Gripper	LPG	**30.0**	8.3	**100.0**	33.3
Gold-m	LPG	**30.0**	29.9	100.0	100.0
M-BW	SatPlan	**26.4**	20.4	**90.0**	73.3
Parking	FF	3.8	**4.6**	13.3	**16.7**
Rovers	LPG	**29.3**	26.1	**100.0**	90.0
Satellite	LPG	28.8	**29.7**	96.7	**100.0**
TPP	Lama	**30.0**	9.1	**100.0**	33.3
All above		**237.8**	172.4	**88.9**	68.1

Table II
TIME/QUALITY IPC SCORE (MAX SCORE 30 PER DOMAIN) AND PERCENTAGES OF PROBLEMS SOLVED BY ASAP AND THE BEST BASIC PLANNER FOR THE SELECTED DOMAINS. BW, GOLD-M AND M-BW STAND RESPECTIVELY FOR BLOCKSWORLD, GOLD-MINER AND MATCHING-BW.

Figure 1 shows results for a two-dimensional comparison, in terms of runtimes, done on the testing problems of Depots. In the top chart we are comparing the CPU times of all the algorithms sharing the same encoding (e) of the selected one, but exploiting different planners. SatPlan is not showed since it does not solve any testing problem. In the bottom chart we are comparing the algorithms sharing the same planner (p) but with different encodings. The impact of both the dimensions considered by the proposed approach is significant in terms of CPU time, and we experimentally observed that it is significant also in terms of quality of the solutions found.

For understanding the usefulness of the knowledge extracted under the form of domain encodings, for verifying the hypothesis that no single planner outperforms all the others in every considered benchmark domain, and for verifying that ASAP effectively selects the most promising algorithm for each selected domain, we have compared ASAP with the best performing basic planner for each considered domain. The best basic planner is exploiting the original domain formulation, and has been selected according to the IPC score achieved on the testing problems. The results of this experiment are shown in Table II. The first interesting result is that both ASAPs and ASAPq have significantly better performance in terms of IPC score and number of solved problems, while considering all the selected domains to-

gether. While analysing the results for every single domain, ASAPs is never worse than the best basic planner, rather it is always better except in Parking, where it selected exactly FF and the original domain as the algorithm to exploit. On the other hand, ASAPq is worse in two of the selected domains; in Parking and in Satellite, mainly due to the smaller number of problems solved by the selected algorithms. Another interesting result that we can derive from Table II is that every considered planner achieves the best Time/Quality IPC score on at least one considered domain, and that the single best planner of a domain is usually different than the one included in the couple selected by ASAP. This means that entanglements and macro-operators have a very significant impact on the planners' performance, and that the impact varies notably from planner to planner.

In order to understand the accuracy of the algorithm selection, we compared the performance of the three best algorithms. In Tables III we present the results of this comparison, in terms of time/quality IPC score, that has been done on the benchmark problems of the selected domains. The performances are shown in terms of IPC score, average CPU time (quality) and solved problems. The * indicates the algorithm selected by ASAP. The mean CPU time/quality are calculated on instances solved by all the three best algorithms of the given domain. We would remark that ASAP selects the most promising algorithm on the basis of the results achieved on the learning problems, while in Tables III the comparison is made by ordering the algorithms on the results that they achieved on the testing instances. Concerning the planners included in ASAP, all of them appear at least once. We can then derive that all the planners are able to efficiently exploit, at least on one domain, the knowledge extracted under the form of different encodings.

Considering the runtime optimization, the planner that appears most frequently in Table III is LPG, followed by Metric-FF and Probe. We can derive that LPG and Metric-FF are the planners that better exploit macro-operators and entanglements for improving runtime. This is quite surprising if we consider that LPG and Metric-FF are the oldest planners included in ASAP, and that they appeared more rarely in Table II. One could argue that, since the plans found by Metric-FF were used for reformulating the domains, the fact that Metric-FF performs well while exploiting entanglements or macro-operators, is not surprising. From this perspective, it is worth noting that LPG is the planner which is able to better exploit this additional knowledge, and that the plans found by Metric-FF were used due to their good quality and to the relatively low CPU time required for finding them. If we focus on the best algorithms for optimising quality of the solutions, the planner which appears most frequently in Table III is again LPG, but in this case it is followed by Lama-11. While LPG is often the best basic solver, as shown in Table II, Lama is the best one only in TPP. It seems then reasonable to deduce that the impact of macro-operators and entanglements is strong on the performance of Lama.

From the point of view of the encodings of the domains, there are no significant differences between runtime and quality, the Outer entanglements appear most frequently in Table III. The less useful encodings are the Inner entangle-

Optimising runtimes				
Domain	**Algorithm**	**IPC**	**Mean CPU**	**% Solved**
Blocksworld	⟨Both, FF⟩*	30.0	1.3	100.0
	⟨Outer, Probe⟩	20.6	3.8	100.0
	⟨Outer, LPG⟩	19.8	8.1	100.0
Depots	⟨Outer, FF⟩*	28.2	0.5	100.0
	⟨Outer, LPG⟩	21.4	0.9	100.0
	⟨Both, LPG⟩	21.2	1.1	100.0
Gripper	⟨Outer, Mp⟩	30.0	8.0	100.0
	⟨Outer, SGPlan⟩*	26.7	11.0	100.0
	⟨Outer, LPG⟩	23.9	15.3	100.0
Gold-miner	⟨Macro, FF⟩*	29.9	0.02	100.0
	⟨Outer, FF⟩	21.8	0.08	100.0
	⟨Inner, FF⟩	18.8	0.1	100.0
Matching-BW	⟨Macro, LPG⟩*	24.6	1.9	100.0
	⟨Both, LPG⟩	21.6	5.4	96.7
	⟨Macro, FF⟩	17.0	42.6	76.7
Parking	⟨Original, FF⟩*	4.6	376.9	16.7
	⟨Inner, Lama⟩	2.6	659.8	13.3
	⟨Original, Probe⟩	2.5	308.7	10.0
Rovers	⟨Macro, LPG⟩*	28.0	45.6	93.3
	⟨Original, LPG⟩	22.0	92.9	93.3
	⟨Outer, LPG⟩	15.1	261.6	86.7
Satellite	⟨Outer, LPG⟩*	23.8	75.9	96.7
	⟨Original, LPG⟩	22.1	92.4	100.0
	⟨Macro, LPG⟩	17.8	85.1	66.7
TPP	⟨Outer, LPG⟩	23.2	34.2	100.0
	⟨Both, LPG⟩*	20.0	14.7	66.7
	⟨Outer, Probe⟩	19.9	61.3	100.0

Optimising Quality				
Domain	**Algorithm**	**IPC**	**Mean Qual**	**% Solved**
Blocksworld	⟨Both, FF⟩*	29.5	231.5	100.0
	⟨Outer, LPG⟩	28.3	243.0	100.0
	⟨Outer, Probe⟩	27.5	248.7	100.0
Depots	⟨Outer, LPG⟩	29.4	120.3	100.0
	⟨Both, LPG⟩*	29.2	120.9	100.0
	⟨Both, Probe⟩	27.7	127.1	100.0
Gripper	⟨Outer, Mp⟩*	30.0	566.8	100.0
	⟨Outer, LPG⟩	28.6	593.8	100.0
	⟨Outer, SGPlan⟩	25.8	659.9	100.0
Gold-miner	⟨Both, SatPlan⟩*	30.0	23.4	100.0
	⟨Macro, Lama⟩	30.0	23.4	100.0
	⟨Macro, LPG⟩	30.0	23.4	100.0
Matching-BW	⟨Outer, SatPlan⟩	26.4	57.2	93.3
	⟨Both, SatPlan⟩	26.4	57.2	93.3
	⟨Outer, LPG⟩*	25.8	61.1	90.0
Parking	⟨Original, FF⟩	4.6	78.0	16.7
	⟨Inner, Lama⟩*	3.8	82.0	13.3
	⟨Inner, FF⟩	2.9	74.5	10.0
Rovers	⟨Macro, Lama⟩*	27.5	674.3	100.0
	⟨Macro, LPG⟩	23.7	669.0	93.3
	⟨Outer, Lama⟩	21.5	657.1	83.3
Satellite	⟨Original, LPG⟩	29.7	675.5	100.0
	⟨Outer, LPG⟩*	28.8	673.1	96.7
	⟨Original, SGPlan⟩	13.7	742.5	50.0
TPP	⟨Outer, Lama⟩*	27.1	422.1	100.0
	⟨Both, Lama⟩	21.0	387.1	73.3
	⟨Outer, Probe⟩	19.7	570.4	100.0

Table III
TIME/QUALITY IPC SCORE (MAX SCORE 30 PER DOMAIN), AVERAGE CPU TIME/PLAN QUALITY AND PERCENTAGES OF PROBLEMS SOLVED BY THE FIRST 3 COUPLES W.R.T. THE IPC SCORE, FOR THE SELECTED DOMAINS. THE * INDICATES THE ALGORITHM SELECTED BY ASAPs/ASAPq.

ments, that rarely allowed algorithms considering them to achieve good results.

In terms of runtimes, ASAPs almost always selects the best algorithm, which usually performs significantly better than the second and the third ones. In two domains, TPP and Gripper, ASAPs selected the second one. It is worth noting that in Gripper domain the performance of the best performing algorithm is similar to the ones of the second. On training problems, their performance were still very close, but the second one had slightly better results. In TPP, the selected algorithm is solving 20 testing problems out of 30. In all of them it is the fastest planner (quality score is exactly 20.0). Interestingly, on the remaining 10 problems, LPG crashed due to memory errors. We believe that this is a bug in the planner and that, without this bug, the algorithm selected by ASAPs would be the best one also in that domain.

In terms of quality, ASAPq selects the best algorithm in five domains; in Depots, Parking and Satellite the selected couple is the second one, in Matching-BW the third one. On the other hand, the three best couples usually achieve similar quality score results, this behavior is quite different between quality and runtimes.

The results shown in Table III indicate that the approach used for selecting the most promising algorithm, even if not very sophisticated, scales well with increasing problem instance size. The couples selected on training instances, easier than testing ones, are achieving very good results also on significantly harder testing instances.

C. ASAP versus PbP2

To evaluate the effectiveness of our approach against the state-of-the-art of learning-based planners, we compared ASAP with PbP2. It is the winner of the learning track of the last IPC, the IPC-7, which was held in 2011. For this comparison we used exactly the same benchmark domains and problems that were used for the IPC-7. PbP2 exploited the same knowledge that it used for the competition, while ASAP was trained on 30 problems, easier than the testing ones, that were generated by using the problem generators provided by the organizers.

Table IV shows performance in terms of time/quality IPC score, mean CPU time (quality) and percentage of solved problems on benchmark problems of the selected domains. The mean CPU time/quality are calculated on instances solved by both the approaches. The mean on all the domains is not indicated because, given the great variability of both CPU time and plan quality across the domains, it is not informative. The results indicate that ASAP performs better than PbP2 in terms of quality of the solution plans, but it achieves slightly worse results in terms of runtimes. ASAPs is significantly faster than PbP2s in four of the selected domains. In particular, ASAPs is significantly faster (more than 2 orders of magnitude) in Depots, where the

Domain	Time IPC		Mean CPU		% Solved	
	ASAPs	PbP2s	ASAPs	PbP2s	ASAPs	PbP2s
Barman	12.0	**30.0**	72.9	**2.0**	100.0	100.0
Bw	**30.0**	16.4	**1.3**	9.9	100.0	100.0
Depots	**30.0**	8.8	**0.5**	76.7	**100.0**	86.7
Gripper	**30.0**	24.7	**11.0**	18.3	100.0	100.0
Parking	3.6	**8.0**	455.3	**172.8**	16.7	**26.7**
Rovers	19.8	**26.2**	58.4	**18.5**	**93.3**	90.0
Satellite	22.9	**30.0**	71.1	**28.3**	96.7	**100.0**
Spanner	15.1	**30.0**	208.1	**15.9**	100.0	100.0
TPP	**20.0**	16.5	**14.7**	113.8	66.7	**83.3**
All	183.4	**190.6**	–	–	85.9	**87.0**

Domain	Quality IPC		Mean Quality		% Solved	
	ASAPq	PbP2q	ASAPq	PbP2q	ASAPq	PbP2q
Barman	29.8	**29.9**	452.8	**449.3**	100.0	100.0
Bw	**29.9**	25.7	**231.5**	269.9	100.0	100.0
Depots	**30.0**	19.2	**118.1**	163.8	**100.0**	86.7
Gripper	**30.0**	28.9	**566.8**	588.7	100.0	100.0
Parking	3.7	**5.0**	82.0	**68.0**	13.3	**20.0**
Rovers	**27.4**	27.3	**693.9**	708.4	100.0	100.0
Satellite	26.8	**28.2**	790.8	**784.3**	96.7	**100.0**
Spanner	30.0	30.0	326.0	326.0	100.0	100.0
TPP	**29.5**	13.4	**343.3**	370.1	**100.0**	50.0
All	**237.1**	207.6	–	–	**90.0**	84.0

Table IV
TIME/QUALITY IPC SCORE (MAX SCORE 30 PER DOMAIN), AVERAGE CPU TIME/PLAN QUALITY AND PERCENTAGES OF PROBLEMS SOLVED BY ASAP AND PbP2 FOR THE SELECTED DOMAINS. BW STANDS FOR BLOCKSWORLD.

entanglements give a great speedup. We noticed that ASAPs is significantly slower than PbP2s in Barman and Spanner. In these domains, our approach was not able to extract any additional knowledge since Metric-FF was not able to solve any training problem, except trivial ones.

Given the fact that some planners (LPG, Metric-FF, Lama and SGPlan) that are included in the ASAP system are also included in PbP2, it is interesting to analyse the domains in which PbP2 selected as a member of the portfolio the planner selected in the algorithm of ASAP. In Barman, both ASAPs and PbP2s are exploiting SGPlan, but PbP2s was able to extract some macro-operators that improve the performance of the planner in that domain. PbP2s configured a portfolio composed of only LPG (without macros) in Rovers, Satellite and Spanner domains. In the same domains, ASAPs also selected an algorithm which included LPG. In all of them PbP2s is faster than ASAPs. It should be noted that in those domains PbP2s exploits a domain-specific configuration of the parameters of LPG, obtained by [22], while ASAPs runs the default configuration of LPG. Since the use of reformulated problems is useful for LPG in such domains (this can be derived by comparing Tables II and III), and considering that PbP2s did not include macros in the corresponding portfolios, we believe that the results achieved in such domains are due to the speedup allowed by the tuned configuration of LPG. On the other hand, in Blocksworld, Depots and TPP ASAPs selected algorithms which include

Metric-FF and LPG, that are available in PbP2s but are not selected; in these domains the performance improvement given by the entanglements is very significant.

In terms of quality of the solution plans, ASAPq achieved better results in five of the selected domains. In two domains, namely TPP and Depots, ASAPq has significantly better results than PbP2q. We also noticed that, counterintuitively, in Rovers, the macros are helpful for improving the quality of the solution plans.

The portfolios configured by PbP2q on the IPC-7 domains are usually composed of either 2 or 3 different planners. It could happen that all the included planners are useful, or that just a subset of them is actually exploited for finding solution plans on testing problems; it is unclear what the real contribution of each planner is to the portfolio. For this reason a comparison between the planners selected by PbP2q and ASAPq is not possible. Only in the Spanner domain PbP2q exploits a portfolio that is composed of a single planner, LPG, without macros. ASAPq selects an algorithm which is composed of LPG and the original domain encoding, which leads the systems to achieve exactly the same results. In the Barman, Depots, Parking, Rovers, Satellite and TPP, the planner included in the algorithm selected by ASAPq is present in the portfolio configured by PbP2q. Finally, in Blocksworld ASAPq is selecting Metric-FF, which is considered in PbP2q but not included in the portfolio, and in Gripper our approach is selecting a planner, Mp, which is not considered by PbP2q.

VI. Discussion

As the results shown in the previous section indicated, an accurate selection of a single algorithm allows ASAP to achieve better results than the portfolio-based planning system PbP2 in terms of plan quality, and to be very close to PbP2 in terms of CPU time. We believe that a domain-specific portfolio can be completely exploited when the different planners can run in *pure* parallel, so when several cores are available. In case a single core is available, using different planners together can slow down the performance of the best one on the given domain.

The algorithms selected by ASAP have shown very good performance on testing problems. As indicated by the results shown in Table III, there is no or very little space for further improvements in terms of CPU time. However, in Matching-BW and Satellite domains the selected algorithms still achieved the best results in total in comparison to the others, but for some problems the results of the selected couple were significantly worse. For instance we observed that in Satellite domain about 40% of testing problems could be solved significantly faster by a different algorithm $\langle \mathsf{Macros}, \mathsf{LPG} \rangle$ then the selected one $\langle \mathsf{Outer}, \mathsf{LPG} \rangle$. It indicates that we need to somehow classify problems even within the same domain. An initial idea assumes that each class of the problem has a different algorithm assigned to it which provides the best results. In the learning stage we can easily identify which couple works best on a particular problem and hence determine classes of problems (the number of classes is equal to the number of algorithms which were best on at least one problem). However, it might happen that some classes will be small (containing only a few problems) which may prevent identification of their characteristic properties. Therefore, it seems to be appropriate to move problems from 'small' classes to larger ones where the corresponding couple is closest to the best. However, an efficient classifier for planning problems has not been developed yet.

Interestingly, the need for classification seems to be restricted to runtime optimization. By analyzing the performance of algorithms, we observed that in terms of plan quality, usually the couple that achieves the best result is finding the best solution for all, or almost all, the testing problems. In the Matching-BW domain the best algorithm is outperformed by a different one only on three problems out of the 30 considered.

Regarding the exploitation of the knowledge extracted in the form of macro-operators and entanglements, we noticed that Metric-FF and LPG are the ones that, in terms of runtimes, exploited it more efficiently. Considering the results shown in Table II, in Blocksworld, Depots, Gold-miner and Matching-BW, Metric-FF and LPG are not the best single planner; but while exploiting the additional knowledge, they outperforms the others, as shown in Table III. Also in terms of quality, the impact of the additional knowledge extracted in the form of new encodings is significant. This is surprising if we consider that macro-operators and entanglements are designed especially for improving the performance of planners in terms of time needed for finding a satisficing solution. A very interesting example of this behaviour is given by Lama in the domain TPP, in which the exploitation of entanglements, either Both or Outer, lets the planner improve its performance by more than the 60%; all the plans found by Lama and the original domain encoding have lower quality.

VII. Conclusion and Future Work

In this paper we have presented ASAP, an automatic algorithm selection approach for planning. ASAP is based on the idea of extracting additional knowledge from a domain, in the form of macro-operators and entanglements, combining such knowledge with existing planning systems for generating new algorithms, and selecting the most promising algorithm for solving problems from the given domain. ASAP has two different versions: ASAPs which selects the most promising algorithm for optimising runtimes, and ASAPq which optimises the quality of the solution plans.

An experimental analysis conducted on a total of 11 well-known benchmark domains and that involved 660 planning problems, has shown that (i) the impact of the considered dimensions on the performances of the algorithms is

significant, (ii) the technique used for selecting the most promising algorithm to exploit on testing problems is very accurate, (iii) ASAPs is competitive with the state-of-the-art of learning-based planning systems, PbP2s, in terms of runtime, (iv) ASAPq outperformed PbP2q in terms of quality of solution plans.

Future work includes further experimental analysis, in particular for understanding if learning-based approaches exploiting domain-specific portfolios would always be outperformed by accurate and efficient automatic algorithm selection based planners, while sharing the same planners and the same additional knowledge, on single core machines. A specific experimental analysis is also needed for having a better understanding of the impact of problems reformulation on the different planning systems; a system for predicting this impact would lead to a great reduction of the learning time needed for selecting the algorithm to use for a specific domain. Moreover, we are interested in combining the approach used for reformulating planning problems with existing techniques for generating macro-operators (e.g., Wizard [16], Macro-FF [1]).

We noticed that the major limitation of ASAP is that, in its current version, it is heavily dependent on Metric-FF. The solutions found by this planner on a small set of training problems are analyzed for extracting additional knowledge. It could happens, and it happened in Barman and Spanner domains, that Metric-FF is not able to solve any non-trivial training problem. To avoid this situation, we are planning to extend the techniques used for extracting macro-operators [5] and entanglements [3] in order to exploit plans produced by different planners. This could also lead to the derivation of more specific additional knowledge that, potentially, could further increase planner performance.

Finally, we are considering including different algorithm selection techniques in ASAP. The current one is mainly based on IPC score, which considers performance and number of solved problems together. The exploitation of more sophisticated score systems could improve the selection accuracy. Alternative selection techniques could be based, for instance, on the well-known PAR10 score, or on statistical analysis.

REFERENCES

[1] 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)* 24:581–621, 2005.

[2] Y. Chen, B. W. Wah and C. W. Hsu, "Temporal planning using subgoal partitioning and resolution in SGPlan", *Journal of Artificial Intelligence Research (JAIR)* 26:323–369, 2006.

[3] L. Chrpa and T. L. McCluskey, "On exploiting structures of classical planning problems: Generalizing entanglements", In *Proc. of the 20th European Conference on Artificial Intelligence (ECAI)*, 240–245, 2006.

[4] L. Chrpa and R. Barták, 'Reformulating planning problems by eliminating unpromising actions', in *Proceedings of SARA 2009*, 50–57, 2009.

[5] L. Chrpa, "Generation of macro-operators via investigation of action dependencies in plans", *Knowledge Engineering Review* 25(3):281–297, 2010.

[6] A. Coles, A. Coles, A. G. Olaya, S. Jiménez, C. L. Linares, S. Sanner and S. Yoon, "A survey of the seventh international planning competition", *AI Magazine* 33:83–88, 2012.

[7] A. Fern, R. Khardon and P. Tadepalli, "The first learning track of the international planning competition", *Machine Learning* 84:81–107, 2011.

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

[9] A. Gerevini, A. Saetti and M. Vallati, "An automatically configurable portfolio-based planner with macro-actions: PbP", In *Proc. of the 19th International Conference on Automated Planning and Scheduling (ICAPS)*, 19–23, 2009.

[10] A. Gerevini, A. Saetti and M. Vallati, "PbP2: Automatic configuration of a portfolio-based multiplanner", In *Booklet of the 7th International Planning Competition*. 2011.

[11] M. Helmert, "The Fast Downward planning system", *Journal of Artificial Intelligence Research (JAIR)* 26:191–246, 2006.

[12] J. Hoffmann, "The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables", *Journal of Artificial Intelligence Research (JAIR)* 20:291–341, 2003.

[13] A. Howe, E. Dahlman, C. Hansen, A. vonMayrhauser and M. Scheetz, "Exploiting competitive planner performance", In *Proc. of the 5th European Conference on Planning (ECP)*, 62–72, 1999.

[14] H. Kautz, B. Selman and J. Hoffmann, "SatPlan: Planning as satisfiability", In *Abstract Booklet of the 5th International Planning Competition*, 2006.

[15] N. Lipovetzky and H. Geffner, "Searching for plans with carefully designed probes", In *Proc. of the 21st International Conference on Automated Planning and Scheduling (ICAPS)*, 2011.

[16] M. A. H. Newton, J. Levine, M. Fox and D. Long, "Learning macro-actions for arbitrary planners and domains", In *Proc. of the 17th International Conference on Automated Planning and Scheduling (ICAPS)*, 256–263, 2007.

[17] S. Richter, M. Westphal and M. Helmert, "Lama 2008 and 2011", In *Booklet of the 7th International Planning Competition*, 2011.

[18] J. Rintanen, "Engineering efficient planners with SAT", In *Proc. of the 20th European Conference on Artificial Intelligence (ECAI)*, 684–689, 2012.

[19] M. Roberts and A. Howe, "Learned models of performance for many planners", In *Proc. of the ICAPS-07 Workshop of AI Planning and Learning (PAL)*, 2007.

[20] M. Roberts and A. Howe, "Learning from planner performance", *Artificial Intelligence* 173(56):536–561, 2009.

[21] J. Seipp, M. Braun, J. Garimort and M. Helmert, "Learning portfolios of automatically tuned planners", In *Proc. of the 22nd International Conference on Automated Planning and Scheduling (ICAPS)*, 2012.

[22] M. Vallati, C. Fawcett, A. Gerevini, H. Hoos and A. Saetti, "Automatic Generation of Efficient Domain-Specific Planners from Generic Parametrized Planners", In *Proc. of the Sixth Annual Symposium on Combinatorial Search (SoCS)*, 2013.