Learning to Improve AI Planning

Smith Kakar

March 23, 2019

Abstract

The field of Automated Planning has significantly advanced over the past decade. Although many powerful domain-independent planners have been developed until now, the efficiency of these planners remained a concern in many domains on various benchmarks. This challenge motivated the researchers to explore and develop various methods including but not limited to machine learning techniques to make the planning process more efficient for a range of domains. This paper is a review of existing techniques that learn macro-operators to improve Automated Planning. The studies also reveal that there is no single domain-independent planner that performs better than all others in every benchmark domain. This has led to design different approaches to combine existing planning algorithms. The paper covers the portfolio-based approach that has a potential to achieve better overall performance than any individual planner. This approach is attracting considerable interest due to its success in the recent International Planning Competitions.

1 Thesis Statement

Since the performance of a planner vary considerably in different domains, many methods exist including but not limited to machine learning techniques to make the planning process more efficient.

2 Introduction

The field of Automated Planning has significantly advanced in the recent years. Many different methods for Automated Planning exist each with their own benefits and limitations in different domains. No one technique is suitable for every domain. There are methods that make efficient planning engines. On the other hand, many methods exist to make the planning process more efficient. Certainly, a number of techniques will usually be required to improve the performance of a planner for a range of domains.

This paper reviews the existing methods that improve the performance of a planner by exploiting information about the domain structure. The automated methods learn this additional knowledge using training problems from the domain and use it to simplify planning for new problems. The review focuses mainly on learning macro-operators [28]. Macro-operators are the sequence of actions combined into one. They can be considered as 'shortcuts' to the goal state. Therefore, it is possible to reach the goal state in fewer steps and hence speed up the planning process.

The rest of the paper is structured according to the following narrative. The next section provides an overview of Automated Planning. Section 4 covers the various methods to improve Automated Planning. In this paper, we have included the methods that learn macro-operators offline and online. These methods specifically deal with the drawbacks of the specific planner. We have also covered the learning methods that generalise to arbitrary planners and domains. The portfolio-based planning method is included in this paper as this approach has a potential to obtain results that we cannot achieve with an individual domain-independent planner. Several different high performance portfolio-based planners have also been developed recently. According to [39], the portfolio-based planning systems have achieved excellent results in the recent International Planning Competitions.

In Section 4.1.1, we provide an introduction to macro-operators and how they can benefit to the Automated Planning. Since the number of instances of macro-operators can possibly be high at run-time, the useful macros that would actually speed up the search are identified using pruning techniques. If pruning is not performed to discard inferior macros, then the branching factor can significantly increase and the induced overhead will potentially outweigh their benefits.

Early contributions to macro-operators are presented in Section 4.1.2. Besides, this section covers the overview of a forward-search heuristics based

state-of-the-art fully automatic planner. Many existing domain-independent planners have been built upon this planner. The section also highlights the drawbacks of using heuristic search based approach.

Furthermore, the methods that learn macro-operators are categorised into offline, online, genetic and parallel methods. We present them in Section 4.1.3, 4.1.4, 4.1.5 and 4.1.6 respectively. Although the methods in last two categories are strictly offline, we described them separately due to their distinct characteristics. The offline methods use component abstraction and/ or solutions of training plans to learn macros from learning problems. On the other hand, online methods learn macros at run-time during search. These methods do not use training problems. The categorisation allows us to compare and contrast these existing techniques effectively and describe how they benefit to the planning process along with their limitations.

Subsection 2 in Section 4 describes an approach to configure and exploit multiple planners to achieve better overall performance than an individual domain-independent planner. We provide a brief overview of the configuration process and mention some existing portfolio-based planners that have achieved impressive results in the recent International Planning Competitions. The last section summarises the paper and draws some conclusions.

3 Background

This section provides the overview of Automated Planning and planning algorithms.

3.1 Automated Planning

This section provides a background of Automated Planning [20]. Automated Planning is a branch of artificial intelligence (AI) that studies the selection of actions by anticipating their outcomes to reach a state that satisfies a number of goals, typically for autonomous intelligent machines. Therefore, the solution of a planning problem is to generate a plan (i.e. a sequence of actions) to reach the goal state (the state that satisfies the desired goals) from the initial state. The technology is being used in simple as well as complex applications such as autonomous robots, controlling space vehicles etc.

Automated Planning can be domain-specific and domain-independent.

The former includes specific representations and techniques adapted to each problem and the latter uses generic representations and techniques. Automated Planning can be carried out offline and online depending upon the environment characteristics. However, most of the approaches assume that there is a single agent and the system is deterministic, static, fully observable and has a finite set of states and a finite set of actions. This is the simplest planning problem known as the classical planning problem. On the other hand, temporal planning deals with actions that run continuously once initiated. Such actions are called *durative* actions. The aim of this planning problem is to minimise the difference between the start and end time of the plan.

Furthermore, probabilistic planning involves the planning problems when the system is not fully observable and actions are non-deterministic with probabilities. Therefore, the objective may not be to reach one of the goal states but that maximises the rewards or minimises the costs. Preference-based planning studies how to generate plans when the objective is not only to reach the goal state but also to satisfy the user preferences.

In domain-independent planning, there is an input domain model with a possible set of actions, initial state and goal states. The domain-independent planners can solve problems from a range of domains. The planning domains and problems are represented using action languages such as PDDL, STRIPS etc. In STRIPS, there is an initial state, a set of goal states and a set of actions along with their pre-conditions (a list of predicates indicating the facts that must be true so the action becomes applicable) and post-conditions (add and delete lists of predicates that indicate the changes in the state). PDDL is a planning domain description language that was developed mainly for the International Planning Competition (IPC).

Henceforth, all the planning models have their own language extensions. Subsequently, there are several planning algorithms for solving the different planning problems. The next section provides an overview of planning algorithms.

3.2 Planning Algorithms

This section includes a brief overview of planning algorithms used for solving different planning problems.

Forward state space search is one of the most widely used algorithms in classical planning. Planning problem is modelled as a state space which is the

set of all possible configurations of a system. Forward search is an algorithm that searches forward from initial state to reach the goal state. The algorithm is mostly enhanced with heuristic function. Heuristics are employed in planning algorithms either to solve the problem quickly or finding an approximate solution to a given problem. The approximation is achieved by trading accuracy, completeness, optimality or precision for the exact solution. In comparison to forward search, backward search algorithm begins with a goal state and searches backward towards the initial state. This method is also called back-propagation.

Many state-of-the-art planners today are based on forward state-space search. Although their excellent performance are associated with domain independent heuristics that perform well across many domains, the planners still fail in many domains where heuristics provide poor guidance. This is explained in more detail under Section 4.1.2. Motivated by these failures, Yoon et al. [43] explored machine learning techniques that identify domain-specific control knowledge to improve or speed-up forward state-space search in a satisficing planning setting. Yoon et al. point out that the resulting learning-based planner is competitive with state-of-the-art non-learning planners across a wide range of benchmark domains and problems.

For probabilistic planning, the value-iteration and policy-iteration methods are used to solve planning problems. One of the new approaches is outlined in [1]. Furthermore, the algorithm portfolio is not a new concept with reference to portfolio-based approach to combine multiple planners. Huberman et al. [26] introduced this method, based on notions of risk in economics, for combining multiple algorithms to run them in parallel. Gomes & Selman [21] also investigated this method by combining several algorithms into a portfolio and running them in parallel. They evaluated this approach on distributions of hard combinatorial search problems and showed the conditions under which it can have a significant computational advantage over traditional methods. Besides, there are several other planning algorithms. The main objective is how to utilise these algorithms effectively to improve the performance of a planner in a range of domains on various benchmarks. In the next section, we present some of the methods that improve Automated Planning.

4 Methods to Improve AI planning

In this section, we will discuss some of the methods that improve AI planning. We will review the existing literature on planners that learn macros and also on portfolio-based planning. Our review covers the similarities and differences of these methods along with their limitations. Simultaneously, we evaluate how these methods make the planning process more efficient.

4.1 Learning by Macro-Operators

This section provides an introduction to macro-operators and reviews existing methods that learn macro-operators and utilise them effectively in order to improve the performance of AI planners.

4.1.1 Macro-Operators

Macro-operators are the sequence of actions combined into one. For example, consider *making a breakfast* before going to the work. This task is composed of several subtasks such as *lay the table*, *cook breakfast* and *serve breakfast*. Macros can be considered as 'shortcuts' to the goal state which means it is possible to reach the goal state in fewer steps. Formally, macro-operators are equivalent to regular operators in that they have a set of variables, a set of preconditions, a set of add effects and a set of delete effects as in [7].

A standard model measures the size of the search space by mainly two parameters - *Branching Factor* and *Search Depth*. The number of instances of macro-operators can be high in general. Subsequently, if macros and the resulting states introduced are not chosen effectively, then the performance of the planner can slow down due to increase in the branching factor. Therefore, it is important to consider that there are more benefits exploiting macros than their drawbacks. Minton [31] has defined this problem as the *utility problem*.

Furthermore, the empirical evidence indicates that the additional information added to the original domain definition should be quite small. There are several methods that minimise the number of macros and their size in the planning problem. For instance, static macro generation in [7] uses many constraints for pruning the space of macro-operators and discards large macros. Besides, macro-filtering techniques discard the unhelpful macros and keeps only the top performing macros for solving future problems.

Learning macro-operators is one of the machine learning techniques to learn search-control knowledge [43] in order to improve the performance of AI planning systems. Once generalisation is achieved by replacing the constants with variables, these macro-operators can be used to speed up the search in future problems.

4.1.2 Early Contributions

Fikes & Nilson [18] made one of the earliest contributions to macro-operators in AI planning. Macro-operators are extracted using previous solution plans. That means only after the solution to a problem became available. Minton [30] explored this method further to filter the generated macro-operators to two types - S-macros and T-macros. S-macros occur frequently in the problem solutions whereas T-macros occur less frequently but can be useful in improving the performance during search.

The research reveals that most of the existing planners are based on heuristic search over the state space. Hoffmann's FF 2.3 [24] is a state-of-the-art fully automatic planner that uses the forward-search heuristics approach. His paper is still considered a seminal work on domain independent planning. This planner is based on the heuristic principle originally developed by Bonet & Geffner [4]. The main idea is to obtain a heuristic estimate by relaxing the task at hand into a simpler task which ignores the delete lists of all operators. The FF planner obtains an explicit solution to the simpler task using a *Graphplan* algorithm [3]. The solution mechanism involves two main phases - *preprocessing* and *search*. Then the goal distance estimate is the number of actions in the relaxed solutions.

These estimates are obtained using a local search strategy called *enforced hill-climbing*. This procedure uses a breadth first search to find a strictly better successor at each intermediate state. FF also employs *helpful action pruning* technique. The main aim of this strategy is to prune the search space using the relaxed plans. Since the useful actions in a state exist in the relaxed plan, therefore it is an efficient heuristic method to consider the successors of any state to only actions that occur in the relaxed solution.

Although the heuristic based planners achieve excellent performance in many domains, their performance depends upon the definition of the heuristic function. This function is computed by considering a simplified version (relaxed problem) of the planning task. However, solving this simplified problem and obtaining the heuristic function are NP-hard problems [41].

Therefore, the solution is achieved by using approximation to calculate the estimated cost to reach the goal. Further, the heuristic functions have their own drawbacks in terms of providing poor guidance to distance to the goal, being computationally expensive and not being able to find solutions with good plan quality in many domains.

The next section reviews one of the methods that not only deal with these drawbacks but also significantly improve the planning performance.

4.1.3 Offline Macro-Action Learning

Most of the existing work on macro-operators is carried out during the offline learning phase. This is the case where solutions of the simple training problems are used to generate the macro-operators.

Many approaches have been developed over the years to eliminate the drawbacks of heuristics highlighted at end of the previous section. Botea *et al.* [5] learn macro-actions in the Macro-FF planner to minimise the negative effects of the heuristics with a significant reduction in search space in many complex domains. They pointed out that "in many domains, the performance of a planner can be improved by inferring and exploiting information about the domain structure that is not explicitly encoded in the initial PDDL formulation".

Their methods follow a common four-step strategy - *Analysis*, *Macro Generation*, *Macro Ranking and Filtering* and *Planning*. Firstly, a planning domain is analysed and new information about the domain structure is extracted, then macro-operators are built based on the acquired information. A filtering and ranking step selects the most efficient macro-operators. Lastly, the selected macros are used to improve planning in new problem instances.

The macro-operators can be generated from the PDDL domain formulations and various training problems. Then, the best macro-operators are added as new operators to the initial domain formulation. Later, the new problem instances along with enhanced domain formulation can be used as input to any planner to solve these instances. Botea et al. [7] defined this method as *CA-ED* which stands for *Component Abstraction - Enhanced Domain*. Component abstraction is a technique performed during analysis stage to form abstract components using static facts (i.e. facts that are true in all the states) in the planning problem.

Knoblock [27] was the first who explored how to automatically generate abstractions for planning domains. His method generates a hierarchy of

abstractions from the initial low-level problem description. The planning algorithm produces an abstract solution which is further refined iteratively to a low-level representation. Bacchus & Yang [2] further improved the abstraction algorithm using problem relaxation. Botea et al. [6] also explored topological abstraction for hierarchical planning. Unlike component abstraction, topological abstraction only uses a single class of static facts with respect to topological relationships in the search space.

The *CA-ED* approach in Macro-FF has certain limitations. Firstly, the component abstraction technique is limited to a small class of problems as it can only be used in domains with static facts in their problem definitions. Secondly, enhancing initial domain with new macro-operators is limited to STRIP domains because of the fact that it is impractical to use the complete macro-operator definitions including precondition and effect formulas in action description language (ADL).

In order to deal with above limitations, Botea et al. [5] have designed and implemented another architecture *Solution - Enhanced Planner* abbreviated as *SOL-EP*. Instead of using component abstraction, the analysis stage in this method involves processing the solutions of training problems. Macros are generated from these solutions and represented as a list of atomic actions instead of combining into a single operator with complete definitions. Since the original domain formulation cannot be enhanced with SOL-EP macros as new operators, this method uses an enhanced planner to handle these macros. This approach was used successfully in the fourth international planning competition (IPC-4). Although this system deals with the drawbacks of specific planning engines (For instance: FF planner), it is less adaptable to different planning engines in general. Henceforth, we will review another technique that improves the speed and coverage of domain independent planning engines.

Chrpa et al. [14] presents a supervised learning technique called MUM that maximises the utility of macro-operators using domain independent constraints for generating macros and limiting the size of the learned macros. The constraints are created through the generation of *outer entanglements* [13]. Outer entanglements are one of the types of entanglements (relations between planning operators and predicates) [9] that are mainly used to restrict the number of planning operator instances. As in with macro-operators, this additional knowledge can be encoded into the domain. Although Chrpa et al. point out that the decision to select outer entanglements is PSPACE-complete in general [10], MUM preserves completeness as outer entangle-

ments extracted using approximation algorithm are applied only to generated macros. Furthermore, macro learning and pruning techniques in MUM are different from the Macro-FF planner.

In comparison to the CA-ED version of Macro-FF planner in which component abstraction is used to learn macros, MUM technique learns macros from training plans by taking into account both adjacent actions and non-adjacent actions. The training plan permutations are computed according to the *action independence* property. This property allows the exchange of adjacent actions in such plans. Macro-FF eliminates complex macros and also limits the number of arguments in macro pre-conditions. Moreover, it selects the n-most used macros in training plans. On the contrary, the number of instantiations of macros in MUM is restricted by using outer entanglements during their generation but there is no limit on the number of macros. However, the macros that can replace some original operators are prioritised over others.

The experiment results in [14] provide insights into the effectiveness of this system over a range of domains and planning engine combinations and also highlight the limitations of this technique where poor quality solutions used in the training phase have led to low performance. The next section presents another method to improve AI planning that generates macros at run-time and has shown the tremendous improvement in the performance on various IPC benchmarks and domains. We will also compare and contrast the techniques employed by this method with offline macro-action learning method.

4.1.4 Online Macro-Action Learning

While Macro-FF learns macro-actions off-line to speed up the searches by reducing the number of evaluated nodes in the problem space, Coles & Smith [17] learn macro-actions on-line in Marvin by using memoisation techniques to escape from plateaus avoiding expensive exhaustive search. In [17], "a plateau is defined to be a region in the search space where the heuristic values of all successors greater than or equal to the best seen so far".

Like Macro-FF, Marvin is also inspired by the state-of-the-art FF planner. It is built upon the Enforced Hill-Climbing (EHC) search strategy algorithm used in FF. The EHC algorithm in FF uses hill-climbing local search using RPG (Relaxed Planning Graph) heuristic to find a strictly-better state. RPG is a *Graphplan* planning graph [3]. If a strictly better successor is not found

in the immediate neighbourhood with a single action step, the algorithm performs an exhaustive breadth-first search in the problem search space to find action sequences leading to a better state.

Alternatively, Marvin uses a least-bad-first search strategy, with helpful actions, for search on plateaux and a greedy best-first search strategy when EHC fails to find a plan. The method learns macro-actions through memoising the plateau-escaping action sequences during search. Once a similar plateaux is encountered again, the learned macros can escape from it and lead to a strictly better state avoiding exhaustive search. However, these macro-actions can no longer be used for solving future problems in the domain.

Furthermore, Coles et al. in [16] present an online technique to store macro-actions for use on new planning problems in the same domain. However, there is a challenge of facing the utility problem with very large collections of macro-operators. Here, this issue is dealt by storing the generated macro-actions as a sequence of unit actions and extracting features from these macro-actions. Macro-pruning is performed on the basis of these features. The features for pruning can be decided on the basis of usage count, instantiation count, length and number of problems solved after last use. However, there is a risk involved with this pruning technique of rejecting the useful macro-actions. We will now compare how the strategies vary with offline and online learning.

Offline learning methods identify the useful macro-actions for future problems by considering the macro-actions that solve the training problems successfully. Training problems are solved with and without using macro-actions. This evaluation reveals whether the particular macro-action has reduced the time taken to solve the problem. It also determines whether the macro-action was used for solving the problem or not.

On the other hand, the success in online macro-action learning relies on the underlying assumption that the use of a macro-action reduces the time taken to solve the problem as this method does not involve solving the training problems prior to planning. Here, the instantiation of each macro-action in a solution plan is counted and then pruning is performed using either *survival of the fittest* or *roulette selection* strategy. The latter strategy is considered better as macro-actions are selected probabilistically without discarding the macro-actions that have not been useful immediately.

The another feature for which offline and online methods adopt different pruning strategies is the instantiation count. Ideally, the ratio of the instan-

tiation count and usage count is considered to keep branching factor in check. In online learning based planners, such as Marvin, pruning is performed after solving every problem which makes this strategy dynamic and online. On the contrary, Macro-FF uses this strategy once after solving five problems in a given domain. The reason for this is the number of macro-actions will decrease monotonically if pruning is performed after solving every problem due to the fact that offline learning does not generate macro-actions during run-time.

Besides, search time pruning techniques can also be employed to select the useful macro-actions for planning. Macro-FF uses a heuristic method called *helpful macro-action pruning*. This is based on the methods that exploit the relaxed graphplan associated with a problem state at search time. One of the methods include *helpful action pruning* as in [24]. However, online learning uses a more strict version of this approach. It is worthwhile to consider that the ordering of macro-actions with respect to the unit actions is significantly important for search efficiency. Macro-FF maintains the order of macro-actions before the unit-actions as it is known before search that these macro-actions are likely provide better performance after running them offline with training problems.

On the contrary, Marvin orders macro-actions after the unit actions in the default configuration in order to use them on plateaux. However, these are not the only two possible strategies, we may want to explore the other possibilities in general. Next, we will discuss the more recent work on online macro-action learning.

The studies reveal that the existing online learning methods are mostly planner-dependent. Therefore, they are less adaptable to different planning engines. Subsequently, there is a need to develop a planner-independent method. Chrpa et al. [12] introduce OMA which stands for Online MAcro generation. This method analyses domain and problem description and also utilises the lessons learnt from existing macro learning techniques for macro-generation.

In this approach, once the macros are generated using above techniques, the problem models are reformulated with extracted knowledge to newer ones. Then, the planning engine is run with these reformulated models. Further the macros are replaced by the corresponding sequences of actions using the solution plan.

The empirical evaluation in [12] shows the performance improvement in a number of domains as it also outperforms some state-of-the-art planners on

various IPC benchmarks. It also points out that OMA is a good alternative method when on offline technique to generate macros is not applicable.

Until now we have reviewed the work on offline and online macro-action learning methods that exploit the knowledge specific to the planners or domains. The next section introduces the generalised method of learning which does not use the explicit knowledge from the planner or domain. The section also highlights the differences in techniques with respect to offline and online macro-action learning methods.

4.1.5 Evolutionary Macro-Action Learning

Although the offline and online learning methods have demonstrated success in improving the performance significantly, they exploit the information specific to the planners or the domains and do not generalise to arbitrary planners or domains. The generalisation is of immense importance as planners and domains are most likely to exhibit different properties with very less in common. Newton *et al.* [33] present an offline evolutionary macro learning method in Wizard that learns macro-actions for arbitrary chosen planners and domains. Generalisation is achieved mainly by using the *genetic algorithm*.

In this case, genetic algorithm is a search heuristic algorithm based on the natural selection. It does not make use of the explicit knowledge about the system. The method learns macros genetically from plans for a given planner, domain and number of example problems. Macros are generated from plans of smaller problems called *seeding problems* and are further evaluated against larger but solvable problems (within given resource limits) called *ranking problems*. They are represented as sequences of inherent actions and as resultant actions generated by regression of the actions in the sequences. This representation is easy to model with PDDL and also makes it convenient to enhance the domain. The fitness of the macros is determined according to the performance of the planners. Further, we will review how this approach varies from offline and online learning.

In addition to the generalisation, this approach measures the performance in terms of time gain. Clearly, this differs from offline Macro-FF planner in that the performance measure is considered to be the number of states explored. As each state might take more time for evaluation, the performance metric in genetic approach is better than the offline learning method even though both these methods evaluate macros by solving training problems.

Moreover, macro generation and learning methods are different from the Marvin planner. As discussed earlier in Section 4.1.4, Marvin generates macros at run-time from plans of relaxed problems found by eliminating symmetries. Additionally, it learns macros to escape a plateaux, which is a planner characteristic, through memoising the plateau-escaping action sequences. However, the genetic approach does not use any properties specific to the planner.

Macro-pruning techniques are used to reduce effort to detect unhelpful macros and are similar to the existing work discussed in offline and online learning methods. Pruning is also performed during macro evaluation with the *augmented domain*. Augmented domain is obtained for each macro by adding it as an additional action to the original domain. If a planner is able to solve a problem using the original domain but not with the augmented domain, then it implies that the macro has caused the overhead. Thus, pruning these macros during evaluation reduces the overall learning time to solve the ranking problems.

The experiment results in [33] clearly indicate the significant improvement in planning with a number of state-of-the-art planners and several domains. One of the main limitations of this method is that the learning time is often very high. Additionally, the studies reveal that most of the existing techniques generate macros from the most frequent action sequences in training plans. The next section presents a new technique that is inspired by resource locking mechanism of critical sections in parallel computing [29].

4.1.6 Learning Critical Section Macro-Operators

This section discusses a very recent work that improves domain-independent planning with macro-operators. In [11], Chrpa & Vallati present a technique that learns macro-operators which capture the whole activities in which a resource is used. These macros aim to capture sequences of operators such that the first operator locks a resource, the intermediate operator uses the resource and the last operator releases the resource. The critical sections in parallel computing can be seen in AI planning with the help of an example as in [11].

"*In Blocks-World, the robotic hand can be seen as a resource. When the robotic hand picks-up or unstacks a block it becomes "locked", that is, no other block can be carried by the robotic hand at that time. When the robotic hand stacks or puts-down the block it is holding, then the hand is "released",*

that is, it can be used to manipulate other blocks."

In contrast to offline learning methods such as MacroFF or MUM, the macro generation in this approach involves capturing the activities in which a resource is locked. However, the macros from offline learning methods can possibly chain the activities to generate more useful macros using iterative approach. The critical selection macros can also be used as original operators during learning phase. Therefore, these macros can be utilised along with offline learning methods in the form of critical section macros or original operators.

The macros are evaluated on several state-of-the-art planners with a range of benchmarks from the International Planning Competition hosted in 2008 and 2011. The results in [11] indicate the usefulness of these macros in some of the domains.

We want to point out that the work discussed so far either increases the efficiency of the planning process by exploiting the domain structure or make planning engines more efficient based on heuristic search. Clearly, there is no single planner that outperforms all the others in every domain. The next section highlights the work on combining multiple planners to achieve better overall performance.

4.2 Learning Portfolios of Planners

In this section, we will review one more method to improve AI planning using a portfolio of planners. This is a growing technique and has a potential to produce impressive results that individual planner cannot achieve in various domains.

This approach has been studied by Gomes & Selman [21]. While there are many powerful domain-independent planners currently available, no individual planner is a clear winner that clearly outperforms all others in every known benchmark domain. This observation motivated researchers to explore techniques to configure and exploit a portfolio of planners to achieve better overall performance. For instance, Xu *et al.* [42] have successfully applied portfolio based approach to SAT (satisfiability problem). Many other researchers have applied this approach in several search domains.

Howe *et al.* [25, 35] made early contributions to configuration and use of portfolio based planners. Further, Gerevini et al. built the domain-specific portfolio with two versions - PbP [19] and PbP2 [38]. PbP planners automatically configure a portfolio of domain-independent planners. The config-

uration is specific to one domain as it is a domain specific portfolio. The family of PbP portfolios have two variants - one focusing on speed and other on plan quality. The selection of planners from the portfolio is based on a statistical analysis comparing the CPU-times and plan qualities for the training problems. The computed planning time slots determine the order of the planners.

Furthermore, PbP generates at-most two alternative sets of macro-actions for each integrated planner using Wizard [33] and at-most five sets of alternative macro-actions with techniques used in MacroFF [5]. During configuration, PbP selects a subset of planners from all the integrated planners on the basis of their performance for a training problem set. Then, it identifies the best cluster of planners and macro-actions for a domain.

There are some more techniques that have recently been applied in this area. The domain-independent portfolio of Fast Downward Stone Soup and its extended version [23, 36] consists of different configurations of a single high-performance planner [22]. There are several different high-performance portfolio-based planners that have been developed recently. Vallati [39] has provided an exhaustive guide to portfolio based planning with reference to existing systems.

Additionally, there can be three different portfolio categories - *Domain-independent*, *Domain-specific* and *Instance-specific*. ArvandHerd [37] is a pure parallel portfolio where it contains four configurations of the random walk planner Arvand [32] (developed by Nakhost & Müller) and one configuration of another domain-independent planner LAMA [34] (developed by Richter & Westphal). ArvandHerd was the winner of the multi-core track in IPC-7 [15]. IBaCoP and IBaCoP2 [8] are two instance based portfolios where the former achieved a runner-up position in the multi-core track and the latter was the winner of the satisficing track of IPC-2014 [40].

For portfolio-based planners, the configuration process is critical to the success. The steps for configuring a portfolio of planners are divided into two sets - Offline and Online configuration, depending upon whether it involves decisions to be taken offline (before working on learning problems) or online (with respect to performance of incorporated planners on learning problems) for configuration. The offline configuration includes the portfolio objective (target), the overall structure, the choice of planners and scheduling strategy for the planners. On the other hand, the online configuration involves the performance measurement, selection, ordering and CPU-time allocation to selected planners. At last, there is a portfolio evaluation step to measure the

performance of configured portfolio on a subset of testing problems.

These systems achieved excellent results in recent International Planning Competitions [39]. Therefore, it is a good step forward to explore these techniques to combine planners effectively.

5 Conclusion

In this paper we have presented a review of techniques that improve Automated Planning. With this research, we have found that the performance of a planner can be substantially improved by exploiting information about the domain structure. We have reviewed some methods that automatically learn this additional knowledge and utilise it to simplify *search* for new problems. The methods that improve Automated Planning fall into some of the categories mentioned as follows:

- Offline Learning - Supervised learning techniques that learn macros using training problems.
  Examples: Macro-FF, MUM
- Online Learning - Methods that learn macros at run-time during search.
  Examples: Marvin, OMA
- Evolutionary Learning - Offline learning methods that learn macros using genetic algorithm and generalise to arbitrary planners and domains.
  Example: Wizard
- Portfolio-based Planning - Methods to configure and exploit portfolios of planners to achieve better overall performance than an individual planner.
  Examples: PbP, PbP2, ArvandHerd, IBaCop, IBaCop2

Furthermore, we pointed out the similarities and differences between these methods including but not limited to the macro-generation, macro-learning and macro-pruning techniques. Notably, there is less published work on online learning methods in comparison to offline learning methods. Moreover, it is evident that some of the methods specifically deal with the drawbacks of the planner but their adaptability to different planning engines is low. Subsequently, we have some other methods that improve the speed and coverage of the different planning engines.

Additionally, the empirical evaluation of these methods with a wide range of domains on IPC benchmarks has provided useful insights into the performance of the planners. This has led to the conclusion that there is no domain independent planner that outperforms all others in every benchmark domain.

As a result, there is a growing trend towards a portfolio-based approach that utilises these powerful techniques and efficient planners to achieve the performance that cannot be obtained by using a single domain-independent planner. Although this approach is interesting, the challenge is how to make a right selection of the planners in the portfolio configuration for any domain which further depends upon the overall objective.

Finally, we would like to point out that the review mainly focused on methods that learn macro-operators to improve Automated Planning. Thus, there may be other learning methods that obtain better results than the ones covered in this paper.

References

[1] Attias, H. Planning by probabilistic inference. In *AISTATS* (2003).

[2] Bacchus, F., and Yang, Q. Downward refinement and the efficiency of hierarchical problem solving. *Artificial Intelligence 71* (11 1994), 43–100.

[3] Blum, A. L., and Furst, M. L. Fast planning through planning graph analysis. *Artif. Intell. 90*, 1-2 (Feb. 1997), 281–300.

[4] Bonet, B., and Geffner, H. Planning as heuristic search. *Artificial Intelligence 129*, 1 (2001), 5 – 33.

[5] Botea, A., Enzenberger, M., Müller, M., and Schaeffer, J. Macro-ff: Improving AI planning with automatically learned macro-operators. *CoRR abs/1109.2154* (2011).

[6] Botea, A., Müller, M., and Schaeffer, J. Extending pddl for hierarchical planning and topological abstraction. In *In Proceedings of the ICAPS-03 Workshop on PDDL* (2003), pp. 25–32.

[7] Botea, A., Müller, M., and Schaeffer, J. Using component abstraction for automatic generation of macro-actions. In *Proceedings*

of the Fourteenth International Conference on Automated Planning and Scheduling (01 2004), ICAPS 2004, pp. 181–190.

[8] Cenamor, I., De La Rosa, T., and Fernández, F. The ibacop planning system: Instance-based configured portfolios. *J. Artif. Int. Res. 56*, 1 (May 2016), 657–691.

[9] Chrpa, L., and McCluskey, T. On exploiting structures of classical planning problems: Generalizing entanglements. In *Frontiers in Artificial Intelligence and Applications: Proceedings of ECAI 2012*, vol. 242. IOS Press, August 2012, pp. 240–245. Paper presented at ECAI 2012, Montpellier, France, 27th - 31st August 2012.

[10] Chrpa, L., McCluskey, T., and Osborne, H. Reformulating planning problems: A theoretical point of view. In *Proceedings of the 25th International Florida Artificial Intelligence Research Society Conference, FLAIRS-25* (2012), pp. 14–19.

[11] Chrpa, L., and Vallati, M. Improving domain-independent planning via critical section macro-operators. In *Proceedings of The Thirty-Third AAAI Conference on Artificial Intelligence* (11 2018), AAAI press.

[12] Chrpa, L., Vallati, M., and McCluskey, T. On the online generation of effective macro-operators. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, Q. Yang and M. Wooldridge, Eds. AAAI Press, July 2015, pp. 1544–1550.

[13] Chrpa, L., Vallati, M., and McCluskey, T. Outer entanglements: A general heuristic technique for improving the efficiency of planning algorithms. *Journal of Experimental and Theoretical Artificial Intelligence* (8 2018).

[14] Chrpa, L., Vallati, M., and McCluskey, T. L. Mum: A technique for maximising the utility of macro-operators by constrained generation and use. In *ICAPS* (2014).

[15] Coles, A., Coles, A., Olaya, A., Jiménez, S., Linares López, C., Sanner, S., and Yoon, S. A survey of the seventh international planning competition. *Ai Magazine 33* (03 2012), 83–88.

[16] Coles, A., Fox, M., and Coles, A. J. Online identification of useful macro-actions for planning. In *ICAPS* (2007).

[17] Coles, A., and Smith, A. Marvin: A heuristic search planner with online macro-action learning. *CoRR abs/1110.2736* (2011).

[18] Fikes, R. E., and Nilsson, N. J. Strips: A new approach to the application of theorem proving to problem solving. In *Proceedings of the 2nd International Joint Conference on Artificial Intelligence* (San Francisco, CA, USA, 1971), IJCAI'71, Morgan Kaufmann Publishers Inc., pp. 608–620.

[19] Gerevini, A., Saetti, A., and Vallati, M. An automatically configurable portfolio-based planner with macro-actions: Pbp. In *ICAPS 2009 - Proceedings of the 19th International Conference on Automated Planning and Scheduling* (10 2009), AAAI press, pp. 350–353.

[20] Ghallab, M., Nau, D., and Traverso, P. Chapter 1 - introduction and overview. In *Automated Planning*, M. Ghallab, D. Nau, and P. Traverso, Eds., The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann, Burlington, 2004, pp. 1 – 16.

[21] Gomes, C. P., and Selman, B. Algorithm portfolios. *Artificial Intelligence 126*, 1 (2001), 43 – 62. Tradeoffs under Bounded Resources.

[22] Helmert, M. The fast downward planning system. *J. Artif. Int. Res. 26*, 1 (July 2006), 191–246.

[23] Helmert, M., and Röger, G. Fast downward stone soup: A baseline for building planner portfolios. *ICAPS 2011 Workshop on Planning and Learning* (01 2011).

[24] Hoffmann, J., and Nebel, B. The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research - JAIR 14* (06 2011).

[25] Howe, A. E., Dahlman, E., Hansen, C., Scheetz, M., and von Mayrhauser, A. Exploiting competitive planner performance. In *Recent Advances in AI Planning* (Berlin, Heidelberg, 2000), S. Biundo and M. Fox, Eds., Springer Berlin Heidelberg, pp. 62–72.

[26] Huberman, B. A., Lukose, R. M., and Hogg, T. An economics approach to hard computational problems. *Science 275*, 5296 (1997), 51–54.

[27] Knoblock, C. Automatically generating abstractions for planning. *Artificial Intelligence 68* (10 2000), 243–302.

[28] Korf, R. E. Macro-operators: A weak method for learning. *Artif. Intell. 26*, 1 (Apr. 1985), 35–77.

[29] Kumar, V. *Introduction to Parallel Computing*, 2nd ed. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[30] Minton, S. Selectively generalizing plans for problem-solving. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence - Volume 1* (San Francisco, CA, USA, 1985), IJCAI'85, Morgan Kaufmann Publishers Inc., pp. 596–599.

[31] Minton, S. Quantitative results concerning the utility of explanation-based learning. In *AAAI* (1988).

[32] Nakhost, H., and Müller, M. Monte-carlo exploration for deterministic planning. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09)* (01 2009), pp. 1766–1771.

[33] Newton, M. H., Levine, J., Fox, M., and Long, D. Learning macro-actions for arbitrary planners and domains. In *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling (ICAPS 2007)*, M. Boddy, M. Fox, and S. Thiebaux, Eds. AAAI Press, California, USA, 2007, pp. 256–263.

[34] Richter, S., and Westphal, M. The LAMA planner: Guiding cost-based anytime planning with landmarks. *CoRR abs/1401.3839* (2014).

[35] Roberts, M., Howe, A., and Flom, O. Learned models of performance for many planners. In *In ICAPS 2007, Workshop AI Planning and Learning* (2007).

[36] Seipp, J., Braun, M., Garimort, J., and Helmert, M. Learning portfolios of automatically tuned planners. In *ICAPS* (2012).

[37] Valenzano, R., Nakhost, H., Müller, M., Schaeffer, J., and Sturtevant, N. Arvandherd: Parallel planning with a portfolio. In *Proceedings of the 20th European Conference on Artificial Intelligence* (Amsterdam, The Netherlands, The Netherlands, 2012), ECAI'12, IOS Press, pp. 786–791.

[38] Vallati, M. Pbp2: Automatic configuration of a portfolio-based multi-planner. In *ICAPS 2011* (2011).

[39] Vallati, M. A guide to portfolio-based planning. In *Multi-disciplinary Trends in Artificial Intelligence* (Berlin, Heidelberg, 2012), C. Sombattheera, N. K. Loi, R. Wankar, and T. Quan, Eds., Springer Berlin Heidelberg, pp. 57–68.

[40] Vallati, M., Chrpa, L., Grzes, M., McCluskey, T., Roberts, M., and Sanner, S. The 2014 international planning competition: Progress and trends. *AI Magazine* (06 2015).

[41] Woeginger, G. J. *Exact Algorithms for NP-Hard Problems: A Survey*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, pp. 185–207.

[42] Xu, L., Hutter, F., Hoos, H. H., and Leyton-Brown, K. Satzilla: Portfolio-based algorithm selection for sat. *J. Artif. Int. Res. 32*, 1 (June 2008), 565–606.

[43] Yoon, S., Fern, A., and Givan, R. Learning control knowledge for forward search planning. *J. Mach. Learn. Res. 9* (June 2008), 683–718.