Lazy Adaptive Multicriteria Planning

Grigorios Tsoumakas, Dimitris Vrakas, Nick Bassiliades and Ioannis Vlahavas

Abstract. This paper describes a learning system for the automatic configuration of domain independent planning systems, based on measurable features of planning problems. The purpose of the Lazy Adaptive Multicriteria Planning (*LAMP*) system is to configure a planner in an optimal way, concerning two quality metrics (i.e. execution speed and plan quality), for a given problem according to user-specified preferences. The training data are produced by running the planner under consideration on a set of problems using all possible parameter configurations and recording the planning time and the plan length. When a new problem arises, *LAMP* extracts the values for a number of domain-expert specified problem features and uses them to identify the k nearest problems solved in the past. The system then performs a multicriteria combination of the performances of the retrieved problems according to user-specified weights that specify the relative importance of the quality metrics and selects the configuration with the best score. Experimental results show that *LAMP* improves the performance of the default configuration of two already well-performing planning systems in a variety of planning problems.

1 Introduction

Domain independent heuristic planning relies on ingenious techniques, such as heuristics and search strategies, to improve the execution speed of planning systems and the quality of their solutions in arbitrary planning problems. However, no single technique has yet proved to be the best for all kinds of problems. Many modern planning systems incorporate more than one such optimizing techniques in order to capture the peculiarities of a wider range of problems. However, to achieve the optimum performance these planners require manual fine-tuning of their run-time parameters.

Few attempts have been made to explain which are the specific dynamics of a planning problem that favor a specific planning technique and even more, which is the best setup for a planning system given the characteristics of the planning problem. This kind of knowledge would clearly assist the planning community in producing flexible systems that could automatically adapt themselves to each problem, achieving best performance.

This paper presents a learning system for dealing with the aforementioned issue. The Lazy Adaptive Multicriteria Planning *LAMP* system automatically configures the parameters (such as search direction and agenda size) of a planner based on measurable characteristics (such as number of actions per operator and mutual exclusions between facts) of planning problems. Learning data are produced by running the planner under consideration off-line on several planning problems using all combinations of values for its parameters. When *LAMP* is faced with a new problem, it retrieves the recorded performance (execution time and plan length) for all parameter configurations of the k nearest problems and performs a multicriteria combination with user-specified weights. The configuration with the best combined score is then used for running the planner with the new problem.

The performance of *LAMP* was evaluated using two state-of-the-art domain independent planning systems and managed to increase their performance in a variety of planning problems. The results also showed that the use of different weights for planning speed and plan length had the expected effect of biasing the planning systems towards optimizing either of the criteria.

The rest of the paper is organized as follows. Section 2 presents related work in Machine Learning and Planning and Instance-Based Learning with multiple criteria. Section 3 describes the problem features and the production of training data for *LAMP*. The next section describes the kNN approach and the multicriteria combination for the prediction of the best configuration. Section 5 presents our adaptation of the Relief algorithm for feature weighting and Section 6 experimental results that prove the significance of *LAMP*. Finally, the last section concludes this work and points areas for improvements.

2 Related Work

Machine learning has been exploited extensively in the past to support Planning systems in many ways. A recent extended survey of past approaches on Machine Learning and Planning can be found in [13].

There are three main categories of approaches based on the phase of planning that learning is applied to and the consequent type of knowledge that is acquired. Domain knowledge is utilized by planners in pre-processing phases in order to either modify the description of the problem in a way that will make it easier for solving it or make the appropriate adjustments to the planner to best attack the problem [7, 11]. Control knowledge can be utilized during search in order to either solve the problem faster or produce better plans [4, 2]. Finally, optimization knowledge is utilized after the production of an initial plan, in order to transform it in a new one that optimizes certain criteria, e.g. number of steps or resources usage [1].

LAMP falls in the domain knowledge category. Compared to relevant approaches, like [11], it has the following advantages: a) it allows the users to specify their priorities in terms of plan length and planning time, b) it can be incrementally trained by running the planner off-line as new problems arise, c) it is orders of magnitude faster in learning due to its lazy nature and only fractionally slower at classification time, and d) apart from the best configuration, it can output a number of alternatives sorted by their score.

The learning approach of *LAMP* follows the paradigm of the work by the METAL European project consortium on Meta-Learning [5]. There, the kNN algorithm was employed to learn a ranking of Machine Learning algorithms based on a multicriteria combination of their accuracy and time measurements [3]. *LAMP* is faced with a similar problem, but has to deal with the much larger size of the feature vector and the increased number of different potential predictions. This motivated us to contribute an investigation of the applicability of the Relief algorithm [9] as a feature weighting method together with the kNN algorithm. In addition *LAMP* uses a different multi-criteria method and applies the kNN algorithm directly on the combined scores of the configurations, instead of using their respective rankings.

3 Collecting the Training Data for *LAMP*

An important decision in the design of any successful learning system is the selection of appropriate experience (training data). For *LAMP* this comes down to the selection of suitable problem features that would correlate with the performance of the different planner configurations. Therefore, a first necessary step that we performed was a theoretical analysis of planning problems, in order to discover salient features that could influence the choice of planning parameters.

Our main concern was to select attributes that their values are easily calculated rather than complex attributes that would cause a large overhead in the total planning time. Therefore, most of the attributes come directly from the PDDL (Planning Domain Definition Language) files, which are the default input to planning systems, and their values can be promptly calculated.

A second concern which influenced the selection of attributes was the fact that they should be general enough to be applied to all domains and their values should not depend so much on the size of the problem. Otherwise the knowledge learned from easy problems would not be applied effectively to difficult ones. For example, instead of using the number of mutexes (mutual exclusions between facts) in the problem as an attribute that strongly depends on the size of the problem (larger problems tend to have more mutexes), we divide it by the total number of dynamic facts and this attribute (mutex density) identifies the complexity of the problem without taking into account the problem size. This is a general solution followed in all situations where a problem attribute depends nearly linearly on the size of the problem.

Taking the above into consideration we resulted in a set of 26 numerical characteristics, which can be divided in two categories: The first category refers to simple and easily measured characteristics of planning problems, e.g. number of actions per operator, that source directly from the input files. The second category consists of more sophisticated characteristics that arise from features of modern planners, such as mutexes or orderings (between goals and initial facts). These characteristics are useful even for the automatic configuration of planners that do not exploit them, since they capture interesting aspects of the problems' morphology. Moreover, there exist very efficient implementations of techniques for sensing their values and the overhead imposed by them is quite low.

Apart from the features of a number of specific planning problems, *LAMP* also requires the performance of the planning system under consideration on these problems using all combinations of values for its planning parameters. For continuous parameters a standard discretization method can be used to obtain discrete intervals. The whole process of recording the training data is illustrated in Figure 1.

For each run of the planner we record the features of the problem, the performance of the planner (steps of the resulting plan and required planning time) and the configuration of parameters. The latter two are recorded straightforwardly as they are the input and output of the planning system. For the efficient calculation of the problem features we developed a problem analyzer that takes as input the PDDL files of the planning problem and outputs the values of the features.

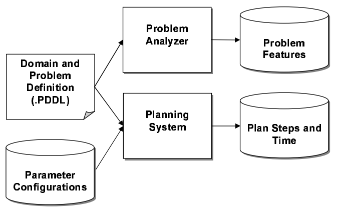


Figure 1. Data collection

The training data were organized as a multi-relational data set, consisting of 2 primary tables, *problems* (M rows) and *parameters* (N rows), and a relation table *performances* ($M * N$ rows), in order to save storage space, enhance the search for the k nearest neighbors and speed up the retrieval of the corresponding performances.

4 Automatic Configuration of Planning Systems

Given a new planning problem p_n, *LAMP* first calculates the values of the problem features using the problem analyzer. Then the kNN algorithm is engaged in order to retrieve the *ids* of the k nearest problems from the *problems* table. In the implementation of kNN we use the Manhattan measure with the normalized values of the problem attributes to calculate the distance between p_n and all problems p_x in the *problems* table:

$$\delta(p_n, p_x) = \sum_f \frac{|p_n(f) - p_x(f)|}{max(f) - min(f)} \tag{1}$$

where f is a problem feature, $p_n(f)$ and $p_x(f)$ the values of this feature for problems p_n and p_x respectively and $max(f)$ and $min(f)$ are the maximum and minimum values of this feature.

The *ids* of the k nearest problems are then used by *LAMP* to retrieve the corresponding plan steps and planning time for all possible configurations in two $k * N$ matrices. The next step is to combine these performances in order to suggest a single parameter configuration with the optimal performance.

Optimal is however susceptible to user preferences, i.e. a shorter plan is usually preferred than a longer one, but there are cases (e.g. real time systems) where the planner must respond promptly even at the expense of the quality of the resulting plan. Since, these two criteria (fast planning, short plans) are contradictory, it is up to the domain experts to set up their priorities. *LAMP* has the advantage of letting the users express their priorities through two parameters: w_s (weight of steps) and w_t (weight of time). The overall planner performance is calculated as a multicriteria combination of the steps and time based on these weights. Specifically, the widely used Weighted Sum method [8] is applied to obtain an overall score from the two criteria, which must first be normalized. For each problem and planner configuration, we normalize time and steps according to the following transformations:

- Let S_{ij} be the number of plan steps and T_{ij} be the required time to build it for problem i (i=1..k) and planner configuration j (j=1..N).
- First, we find the shortest plan and minimum planning time for each problem among the tested planner configurations:

$$S_i^{min} = \min S_{ij} \qquad T_i^{min} = \min T_{ij}$$

- Then, we normalize the results by dividing the minimum plan length and minimum planning time of each run with the corresponding problem value.

$$S_{ij}^{norm} = \frac{S_i^{min}}{S_{ij}} \qquad T_{ij}^{norm} = \frac{T_i^{min}}{T_{ij}}$$

- Subsequently *LAMP* calculates an overall score as the average of the normalized criteria weighted by the user-specified weights:

$$Score_{ij} = w_s * S_{ij}^{norm} + w_t * T_{ij}^{norm}$$

Finally *LAMP* averages the planner configuration scores across the k nearest problems and outputs the one with the largest average. The whole process is illustrated in Figure 2.

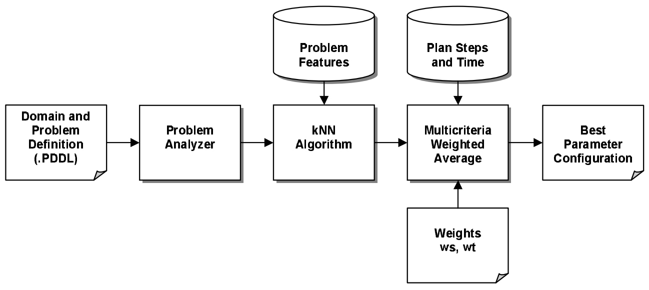

Problem Features
Plan Steps and Time
Domain and Problem Definition (.PDDL)
Problem Analyzer
kNN Algorithm
Multicriteria Weighted Average
Best Parameter Configuration
Weights ws, wt

Figure 2. Predicting the best parameter configuration for a problem

4.1 Advantages and Disadvantages

A very important advantage of *LAMP* is that can be trained incrementally with each new planning problem that arises, by running off-line the corresponding planning system. This makes it possible to constantly enrich *LAMP* with new problems and thus enhance its predictive performance.

In addition, it is worth noting that *LAMP* can actually output a vector of average scores for all parameter configurations instead of a single parameter configuration. This can be exploited for example in order to output the top 10 configurations and let the user decide amongst them. Another useful aspect of the ordering of the configurations for non time-critical applications is to run the planner with the top performing configurations sequentially or in parallel and obtain the smallest plan.

An important advantage of *LAMP* is the fact that users can input their priorities for steps and time with the corresponding weights. As it is shown from the experimental results, setting the weights has the desired effect on the planner performance. This allows the users to customize the planning system towards the optimization of either of the criteria.

Finally, *LAMP* does not require any training time. If training was required then for each different set of multicriteria weights, a different model would have to be prepared. This would increase the complexity of the approach and lead to an inflexible system.

The disadvantages of *LAMP* are increased storage needs and time during prediction. The system must locate the k nearest problems of a new problem, which requires a scan of all records in the problem table and calculation of distances. For our experiments, the size of the problem table was small enough to fit in main memory. Furthermore, there exist efficient indexing techniques for reducing the complexity of kNN queries in large databases, should the problem base becomes large. In any case the classification time is a fraction of the total planning time for real-world planning problems.

5 Feature Weighting

A known limitation of lazy learning algorithms of the k-nearest neighbor (kNN) family is their sensitivity to irrelevant features. Wettschereck et. al [12], argue that feature weighting methods tend to outperform feature selection algorithms for tasks where some features are useful but less important than others. In order to clarify the utility of the domain-expert engineered features of planning problems we have followed a feature weighting approach. In the same study it has been shown that feature weighting methods that use performance feedback to assign weight settings require less preprocessing, perform better in the presence of interacting features and generally require less training data to learn good settings. In our case, where training data (planning problems) are not expected to be that many and where there could be interaction amongst the problem features, we decided to follow a performance feedback approach.

Specifically, *LAMP* performs feature weighting based on the Relief family of algorithms which was introduced by Kira and Rendell [9] and extended by Kononeko [10]. In its original formulation, Relief starts by assigning all feature weights to 0. It then loops over a random sample of the data set and for each instance x locates the most similar positive (p) and negative (n) instances, called *nearest hit* and *nearest miss* respectively. Based on these instances the weight of each feature is updated using the following formula:

$$w(f) = w(f) - \delta(x(f), p(f)) + \delta(x(f), n(f)) \qquad (2)$$

The problem with the application of Relief to our data is how to determine whether two instances belong to the same class, or not, as each instance is associated with a vector of scores for the configurations. To deal with this problem we employed *Pearson's product-moment correlation coefficient* (r) to measure the linear correlation of two such vectors X, Y of size n:

$$r_{XY} = \frac{n\sum XY - (\sum X)(\sum Y)}{\sqrt{[n\sum X^2 - (\sum X)^2][n\sum Y^2 - (\sum Y)^2]}} \qquad (3)$$

This returns a number between -1 and 1, where 1 indicates positive correlation, -1 negative correlation and 0 no correlation. Given a vector and a confidence interval, we considered another vector a *hit* if their correlation was higher than a critical value that is calculated based on the degrees of freedom (i.e. the size of the vector minus two) and a *miss* otherwise. In the implementation of Relief, we used the 10 nearest hits and misses to update the weights, as suggested in [10], and we processed each training instance of the data set exactly once.

6 Experimental Results

For the evaluation of the learning performance of *LAMP*, an actual domain-independent planning system along with the descriptions of several planning problems are required in order to collect the necessary data. The next two subsections deal exactly with these issues. The first describes the planning systems and the following the planning problems that were used.

The last subsection evaluates a) the usefulness of *LAMP* in boosting the performance of the planning systems, b) the feature weighting approach and c) the effect of the multicriteria weights to the performance of the planner in terms of plan steps and planning time.

6.1 The Planning Systems

For the evaluation of *LAMP* we used LPG [6] and HAP [11], two publicly available state-of-the-art planning systems from two different research groups working on domain-independent Planning. LPG is a planning system that performs stochastic local search in temporal Action Graphs. It can be customized through a number of parameters, but for the purposes of this research we selected the 4 with the maximum impact on the performance of the planner, which are outlined in Table 1. HAP is a planning system that performs a classical search in the space of states. It uses various heuristic mechanisms in order to enhance this search. The system can be configured through the 7 planning parameters that are outline in Table 1.

Table 1. The planning parameters of LPG and HAP and their value sets

Planner	Parameter	Value Set
LPG	Heuristic	{1, 2}
	Restarts	{25, 50, 75}
	Search Steps	{100, 500, 1000}
	IChoice	{1, 2, 3, 4}
HAP	Direction	{0, 1}
	Heuristic	{1, 2, 3}
	Weights (w_1 and w_2)	{0, 1, 2, 3}
	Penalty	{10, 100, 500}
	Agenda	{10, 100, 1000}
	Equal Estimation	{0, 1}
	Remove	{0, 1}

6.2 The Planning Problems

For the production of training data we run LPG and HAP using all possible parameter configurations on a total of 450 planning problems, which correspond to 30 planning problems from each of the 15 planning domains, outlined in Table 2. Some problems were not solved by any configuration of LPG and HAP. These were excluded from learning as they don't offer any information. The actual number of problems per domain used for training *LAMP* for each planner are given in the last two columns of Table 2.

Table 2. Planning domains, their sources and the number of problems solved by at least one configuration for each of the two planners

Domain	Source	LPG	HAP
Assembly	New domain	30	29
Blocks-world (3 operators)	Bibliography	30	30
Blocks-world (4 operators)	AIPS 98, 2000	30	30
Driver	AIPS 2002	30	30
Ferry	FF collection	28	28
Freecell	AIPS 2000, 2002	9	30
Gripper	AIPS 98	30	30
Hanoi	Bibliography	17	28
Sokoban	New domain	15	28
Logistics	AIPS 98, 2000	30	30
Miconic-10	AIPS 2000	28	30
Mystery	AIPS 98	29	30
Tsp	FF collection	30	30
Windows	New domain	30	30
Zeno	AIPS 2002	29	30
	Total	395	443

6.3 Evaluation

For comparison purposes we measured the average performance of all different configurations of the two planners on all the above problems. The configurations with the best scores served as a baseline for comparison with the score of the automatic configuration of the planners based on the predictions of *LAMP*. Table 3 reports the scores and the parameters of the best configurations for the two planners using three pairs of weights for steps and time: a) w_s=1, w_t=1, b)w_s=2, w_t=1, c) w_s=1, w_t=2.

Table 3. Scores and parameters of the best configurations for the two planners using three pairs of weights for steps and time

w_s	w_t	H	R	S	I	LPG Score	D	H	W	P	A	E	R	HAP Score
1	1	2	50	1000	2	1.49	0	1	2	10	100	1	0	1.54
2	1	2	75	100	2	2.32	0	1	2	10	100	1	0	2.40
1	2	2	50	1000	2	2.18	0	1	2	500	10	0	0	2.23

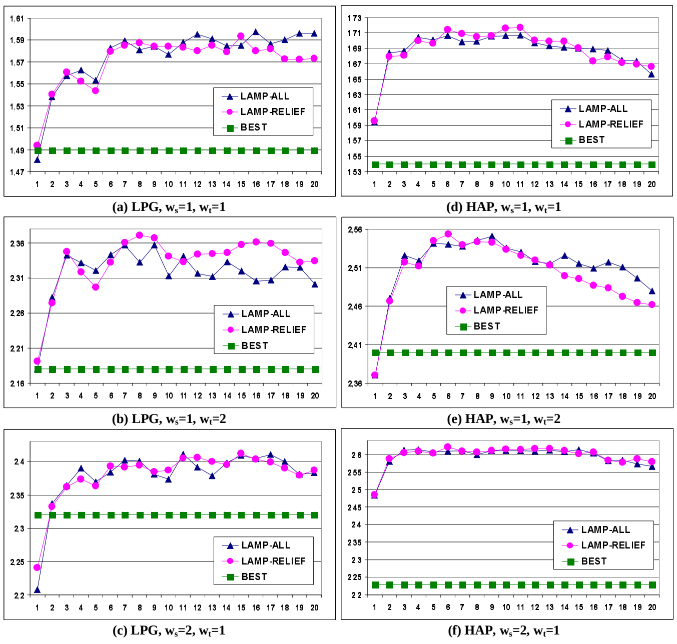
Note that the default configurations of the two planners have a smaller average score than the best configurations, because they are not tuned for any specific weight settings or problems. A fairer comparison would perhaps involve the default configurations of the two planners. For example, when these planning systems participate in international planning contests they are submitted with their default configurations. However we wanted to benchmark *LAMP* against the best performance that these planning systems could achieve with any manual configuration taking into account the corresponding multi-criteria weights.

Cross-validation was used to accurately evaluate the automatic configuration of LPG and HAP by *LAMP*. Specifically the original sets of problems (395 for LPG and 443 for HAP) were split into 10 problem sets of equal size. The training set for each fold was used to calculate the feature weights using our adaptation of Relief and to find the k nearest problems for each of the problems in the test set, using a) the weights selected by Relief, b) all weights equal to 1. The above-described experiment was run for the three pairs of multicriteria weights that were also used for the evaluation of the different configurations of the planners. Figure 3, shows the average scores of *LAMP* using the above specified feature and multicriteria weights and the best configurations of LPG and HAP planners.

A first conclusion stemming from the experimental results is that *LAMP* manages to achieve a good boost in performance compared to that of the best configurations in Table 3, especially for values of k greater than 3. For example, using k=15 and LPG, we gained 6% and using k=6 and HAP we gained 12% in performance for the three weight settings on average.

The results show that *LAMP* manages to achieve better performance in the automatic configuration of HAP than of LPG. A reason behind this could be the fact that the training data for HAP are 10% more than that of LPG. Another reason is that HAP with 7 parameters is more flexible than LPG that only has 4 parameters to tune.

A related noticeable trend is that the performance of LPG tends to increase for larger values of k, while for HAP it decays for values of k larger than 10. If we accept that problems of the same domain are usually closer to each other than problems of other domains (which is verified by the data) then we can interpret this trend as follows. Perhaps *LAMP* requires more neighbors for LPG than for HAP, because for certain domains (see Table 2 there are much fewer problems solved by LPG than for HAP and thus more nearest problems



(a) LPG, w_s=1, w_t=1 (d) HAP, w_s=1, w_t=1

(b) LPG, w_s=1, w_t=2 (e) HAP, w_s=1, w_t=2

(c) LPG, w_s=2, w_t=1 (f) HAP, w_s=2, w_t=1

Figure 3. Average score of LPG and HAP with *LAMP* and their best confi gurations

are required to find good results.

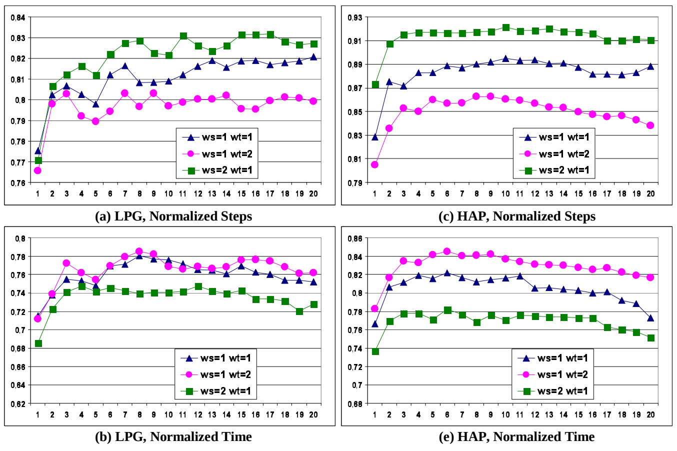
A second conclusion from the experimental results is that Relief manages to increase the performance of *LAMP* on average, but it does not make a statistically significant contribution. This probably means that Pearson's product-moment correlation coefficient is not a suitable statistic for this kind of problems. The truth is that the large number of degrees of freedom (70 for LPG and 862 for HAP) make the coefficient unreliable, especially as it is insensitive to whether the differences of the vectors concern the best performing configurations, that actually influence the selection of the best configuration, or not.

Figure 4 compares the average normalized score of steps and time for the three different weight settings (w_s, w_t) for all nearest neighbors. An important conclusion from these graphs is that setting the weights in favor of planning speed or plan quality has the desired effect on the performance of the planners. We notice that normalized steps are increased for larger weight to steps and decreased for larger weight to time. The same applies for normalized time.

7 Conclusions and Further Work

This work combines two important areas of Artificial Intelligence. It utilizes Machine Learning to adapt domain independent Planning systems to a) the given planning problem that they have to solve and b) the preferences of the users in terms of plan quality and execution speed. The experimental results showed that *LAMP* generalizes successfully from past runs of at least two well-performing planning systems and manages to boost their performance on a variety of planning problems.

In the future we plan to find paths to further improve the performance of *LAMP*. A first thing to research into is an effective feature weighting or feature selection method, taking into account the vector of scores. A related line of research involving Knowledge Engineering for Planning is the extraction of more informative features from the planning problems. Finally we intend to investigate the effect of



(a) LPG, Normalized Steps (c) HAP, Normalized Steps

(b) LPG, Normalized Time (e) HAP, Normalized Time

Figure 4. Normalized steps and time of LPG and HAP

a weighted distance kNN approach.

REFERENCES

[1] J.L Ambite, C.A Knoblock, and S. Minton, 'Learning Plan Rewriting Rules', in *Proceedings of the 5th International Conference on Artificial Intelligence Planning and Scheduling Systems*, eds., S. Chien, S. Kambhampati, and C. A. Knoblock, pp. 3–12. AAAI Press, (2000).

[2] D. Borrajo and M. Veloso, 'Lazy Incremental Learning of Control Knowledge for Effi ciently Obtaining Quality Plans', *Artificial Intelligence Review*, **10**, 1–34, (1996).

[3] P.B. Brazdil, C. Soares, and J.-P. Da Costa, 'Ranking Learning Algorithms: Using IBL and Meta-Learning on Accuracy and Time Results', *Machine Learning*, **50**, 251–277, (2003).

[4] J. Carbonell, C. A. Knoblock, and S. Minton, *PRODIGY: An integrated architecture for planning and learning*, volume K. VanLehn, ed., chapter Architectures for Intelligence, 241–278, Lawrence Erlbaum Associates, 1991.

[5] METAL consortium. Esprit project metal. www.metal-kdd.org., 2002.

[6] A. Gerevini and I. Serina, 'Lpg: a planner based on local search for planning graphs with action costs', in *Proceedings of the 6th International Conference on AI Planning and Scheduling (AIPS'02)*, pp. 13–22, (2002).

[7] A. Howe and E. Dahlman, 'A critical assessment of Benchmark comparison in Planning', *Journal of Artificial Intelligence Research*, **1**, 1–15, (1993).

[8] C.L. Hwang and K. Youn, *Multiple Attribute Decision Making - Methods and Applications: A State of the Art Survey*, Springer-Verlag, New York, USA, 1981.

[9] K. Kira and L. A. Rendell, 'A practical approach to feature selection', in *Proceedings of the 9th International Conference on Machine Learning*, pp. 249–256, (1992).

[10] I. Kononenko, 'Estimating attributes: Analysis and extensions of relief', in *Proceedings of the 1994 European Conference on Machine Learning*, pp. 171–182, (1994).

[11] D. Vrakas, G. Tsoumakas, N. Bassiliades, and I. Vlahavas, 'Learning rules for Adaptive Planning', in *Proceedings of the 13th International Conference on Automated Planning and Scheduling*, pp. 82–91, Trento, Italy, (2003).

[12] D. Wettschereck, D. W. Aha, and T. Mohri, 'A Review and Empirical Analysis of Feature Weighting Methods for a Class of Lazy Learning Algorithms', *Artificial Intelligence Review*, **11**, 273–314, (1997).

[13] T. Zimmerman and S. Kambhampati, 'Learning-Assisted Automated Planning: Looking Back, Taking Stock, Going Forward', *AI Magazine*, **24**(2), 73–96, (2003).