Improving relaxed planning graph heuristics for metric optimization *

Raquel Fuentetaja and **Daniel Borrajo** and **Carlos Linares**
Departamento de Informática, Universidad Carlos III de Madrid
Avda. de la Universidad, 30. 28911 Leganés(Madrid). Spain
rfuentet@inf.uc3m.es, dborrajo@ia.uc3m.es, clinares@inf.uc3m.es

Abstract

Currently a standard technique to compute the heuristic in heuristic planning is to expand a planning graph on the relaxed problem. This paper presents a new approach to expand the planning graph, such that heuristic estimations are more accurate when an optimization metric criteria is given. Additionally, a new kind of Hill-Climbing search that combines two heuristics is proposed to deal with metrics. Results show that the quality of the resulting plans with respect to the metric optimization criteria is improved.

Introduction

One of the most successful techniques in STRIPS planning is to guide the search through the state space by a heuristic function derived from the specification of the planning domain and problem. One general approach for deriving heuristics is to formulate a simplified (or relaxed) version of the problem. Solving this simplified problem is, in general, easier than solving the original problem. Thus, the solution of the relaxed problem is used to estimate the distance to the goal. The most common relaxation used by the planning community, first proposed by McDermott (McDermott 1996) and used in a big number of planners is to ignore the delete actions of the domain operators. Examples of these planners are HSP (Bonet & Geffner 2001), FF (Hoffmann & Nebel 2001), GRT (Refanidis & Vlahavas 2001), and SAPA (Do & Kambhampati 2003). A usual technique to generate the relaxed plan is the application of GRAPHPLAN (Blum & Furst 1995) to the relaxation. The GRAPHPLAN's planning graph can be generated in polinomial complexity (both in time and space) under such a representation where propositions have binary domains.

Nowadays, some heuristic search planners have been extended to deal with representations that express more faithfully features of real-world problems, like temporal actions and numerical resource requirements. The most widely accepted language to model these features is PDDL2.1 (Fox & Long 2003), which was used as the input language for the 3rd International Planning Competition (Long & Fox 2003). Among other features, PDDL2.1 incorporates the possibility to define numerical constraints and effects on a finite set of numerical state variables. These numerical state variables have been typically used to represent fuel consumption, distances, data capacities, etc. Also, PDDL2.1 allows defining problems with quality metrics. A quality metric is usually a numerical expression defined in terms of numerical state variables to be either maximized or minimized. Therefore, a solution plan is said to be optimal if the sum of operators costs is maximal or minimal with respect to a given quality metric.

When dealing with domains with numerical state variables, where the domains of these variables are usually real numbers, the following items shall be carefully observed:

1. the pre-conditions of actions may impose complex constraints (inequalities) on these variables, such as `(fuel plane)>100`.
2. the effects of actions may modify the values of the variables in several ways, such as `increase (fuel plane) in 10`, `decrease (fuel plane) by 10`, etc.
3. the objective of the planner is not just to find a short plan in terms of number of operators in it, but to find an optimal (or near optimal) plan with respect to the metric expression.

In general, the way a numerical heuristic planner solves these issues determines the values of the heuristic estimation and, therefore, the search tree and the quality of the generated plans. The idea presented in this paper attempts to improve Metric-FF (Hoffmann 2003) operation in item (3). Metric-FF is an extension of the FF (Hoffmann & Nebel 2001) planner to numerical state variables. Metric-FF considers the numeric conditions of the actions (1) in the heuristic estimation, but ignores the decreasing effects (2). Regarding to (3), Metric-FF expands the same planning graph (the traditional one, in which a level of propositions is followed by a level with all applicable actions given these propositions) whether when a metric expression is defined or not. In the former, the heuristic cost of a state is the summed up cost of the actions in the respective relaxed plan. In the latter, the heuristic cost of a state is the number of actions in the relaxed plan.

*This work has been partially supported by the Spanish MCyT project TIC2002-04146-C05-05, MEC project TIN2005-08945-C06-05 and regional CAM-UC3M project UC3M-INF-05-016.

In this paper, we propose a technique to expand the planning graph when a metric expression is defined. The aim is to improve the heuristic estimation extending the planning graph by levels of costs instead by levels of operators. The claim is that using more accurate heuristic estimations potentially leads to solutions with better quality, even if, as in the case of Metric-FF, the heuristics are non-admissible. Additionally, we propose to apply a new scheme of Hill-Climbing search to deal with metric optimization in numerical domains.

Relaxed graphplan as a heuristic estimator

The idea of relaxing a planning task consists of solving, in any search state, a relaxed task, and take the length of the relaxed solution as an estimate of how long the solution from the state at hand is. This section describes the generation of the relaxed planning graph applied by Metric-FF. The basic algorithm is shown in Figure 1 (for the sake of simplicity the algorithm shown does not include the handling of numerical features).

```
$t = 0$;
$P_0 = s_0$;
while $G \nsubseteq P_t$ do
    $A_t = \{a \in A \mid prec(a) \subseteq P_t\}$
    $P_{t+1} = P_t \cup add(a), \forall a \in A_t$
    if $P_{t+1} = P_t$ then fail endif
    $t = t + 1$
endwhile
final_layer= $t$, $succeed$
```

Figure 1: Classical graphplan expansion for relaxed planning tasks.

The planning graph in the relaxed case is simply represented as a sequence P_0, A_0, …,P_{t-1}, A_{t-1}, P_t of proposition sets and action sets. These are built incrementally starting with $P_0 = s_0$ as the initial layer, and iteratively inserting the add effects of all applicable actions. The algorithm fails if at some point before reaching the goals no new propositions were inserted. This happens when the relaxed task is unsolvable.

States where the relaxed planning graph does not reach the goals have an infinity heuristic value. Otherwise, once the graphplan is expanded, Metric-FF starts a relaxed plan extraction mechanism to extract a sequential relaxed plan. This mechanism is heuristically guided to obtain short solutions. The length of the optimal sequential relaxed plan is an admissible heuristic for STRIPS tasks. Actually, the relaxed planning graph contains this optimal relaxed solution, but it cannot be synthesized efficiently (Hoffmann & Nebel 2001). Instead, the algorithm extracts a provably suboptimal relaxed plan so that the heuristic estimation obtained is non-admissible.

When this algorithm is used to estimate heuristics in numeric domains, the metric expression is always transformed into a minimizing expression. Metric-FF uses the same planning graph expansion, but the heuristic measure of each state is the summed up cost of all actions in the relaxed plan. As in the previous case, this heuristic estimation is provably non-admissible.

Relaxed graphplan levelled by costs

This section introduces our proposal for expanding the planning graph with metrics. The rationale is to include information about the action costs according to the metric. The relaxed planning graph expansion outlined in the previous section together with the plan extraction mechanism provides an estimation of the length of the shortest solution from the state at hand. Therefore its application seems adequate when one prefers to find a solution minimizing the number of actions in it. But, when a metric expression is given, previous algorithm provides as heuristic estimation the summed up costs of all actions in a relaxed solution obtained from a planning graph built to minimize plan length. This heuristic could be very far of the real cost of the optimal solution with respect the given metric. This happens when the optimal solution regarding the metric has more actions than the optimal solution regarding the plan length.

The method we propose tries to achieve a more informative (though non-admissible) heuristic for problems with a quality metric. The idea is to expand the planning graph in an order such that a sequential relaxed solution near the optimal regarding the metric can be extracted from it. Therefore, our proposal is to modify the graph plan expansion algorithm, but to maintain the same relaxed plan extraction mechanism as Metric-FF. Thus, our heuristic estimation will be the summed up cost of the actions in the sequential relaxed plan extracted too. As in the case of Metric-FF this heuristic is non admissible.

As in the traditional case, the levels of propositions and actions are built incrementally. But, the action levels in the graph are labelled with a cost indicating the maximum cost that can be spent until each level. Thus, we delay in each action level all applicable actions whose costs will lead to a situation in which the corresponding level cost limit is exceeded. Therefore, a set of the delayed actions is maintained in each action level. Figure 2 shows the algorithm.

In this algorithm:

- $\triangle cost_t$ represents the cost increment between action level t and the previous action level $(t-1)$
- $cost\_limit_t$ represents the maximum cost spent until level t. It is computed as the sum of all $\triangle cost$ until level t, and
- D_t is a set containing the delayed actions at level t

From the initial layer $P_0 = s_0$, the algorithm iteratively adds the effects of all applicable actions that hold a) or b) are inserted:

a) The action does not belong to any delayed set and its cost is equal or smaller than the cost increment in the actual level.

b) The action has been previously delayed, and its cost is equal or smaller than the cost limit in the actual level minus the cost limit where the action was delayed.

If a non-delayed action, applicable in level t, does not hold a), it is included in the delayed actions set of level t.

```
$t = 0$;
$P_0 = s_0$;
$D_0 = \emptyset$;
$cost\_limit_{-1} = 0$;
while $G \not\subseteq P_t$ do
    $\triangle cost_t = compute\_cost\_increment()$;
    $cost\_limit_t = cost\_limit_{t-1} + \triangle cost_t$;
    $A_t = \{a \in A \mid prec(a) \subseteq P_t \wedge [(\neg \exists t', a \in D_{t'} \wedge cost(a) \leq \triangle cost_t) \vee (\exists t', a \in D_{t'} \wedge cost(a) \leq cost\_limit_t - cost\_limit_{t'})]\}$
    $D_t = \{a \in A \mid prec(a) \subseteq P_t \wedge (\neg \exists t', a \in D_{t'} \wedge cost(a) > \triangle cost_t)\}$
    $D_{t'} = D_{t'} - A_t, \forall t'$
    $P_{t+1} = P_t \cup add(a), \forall a \in A_t$
    if $P_{t+1} = P_t \wedge D_{t'} = \emptyset, \forall t'$ then fail endif
    $t = t + 1$
endwhile
$final\_layer = t, succeed$
```

Figure 2: Proposed graphplan expansion for relaxed planning tasks for metric optimization.

When an action delayed at level t' holds b), it is removed of the delayed actions set of level t'.

The $\triangle cost$ in each level is computed by the function $compute\_cost\_increment()$. We have implemented three versions of this computation:

v1: $\triangle cost_t$ is fixed in all the planning graph and it is the cost of the cheapest action in the problem instance with cost greater than zero.

v2: $\triangle cost_t$ is computed as the minimum cost of all applicable actions in level t with cost strictly greater than zero.

v3: $\triangle cost_t$ is computed as minimum cost of all applicable actions in level t with cost equal or greater than zero.

Now, the algorithm fails if at some point before reaching the goals no new propositions come in, and there are not delayed actions. As in the classical case, when the algorithm fails the relaxed task is unsolvable, and therefore the heuristic takes an infinity value.

A similar approach to build a planning graph levelled by costs has been used by (Sapena & Onaindía 2004). The main difference between their work and ours is that they propagate in the graph costs of actions as the sum of the costs of their preconditions, and costs of propositions as the cost to reach the action that achieves each proposition plus the cost of the action. We think that this kind of propagation is not needed because the cost to reach each action and proposition can be estimated using the $cost\_limit$ of the corresponding level.

Search algorithms

This section introduces the search algorithms used by Metric-FF, together with our own proposal about what algorithm to use and how it could be adapted to deal with optimization metrics.

Metric-FF uses the heuristic estimation in a forward state search. When an optimization criteria is given, Metric-FF uses a standard weighted A*. However, the search scheme used by Metric-FF for solving problems without an optimization criteria is a kind of Hill-Climbing search called *Enforced Hill Climbing* (EHC). EHC starts in the initial state and performs a number of search iterations trying to improve the heuristic value, until a state with zero value is reached. EHC uses a complete breadth first search to find a strictly better, possibly indirect, successor. The search prunes states that have been seen earlier during the same iteration, and does not expand states that the heuristic function recognizes as dead ends. Besides, EHC uses a pruning technique: selecting a set of the most promising successors to each state. The unpromising successors can then be ignored (thus, it is not a complete algorithm). A promising successor is a state generated by a *helpful* action, where helpful means that it achieves at least one of the goals needed to build the relaxed plan (in the relaxed plan extraction algorithm) in the second level of propositions of the planning graph (the first level is the initial state). Therefore, if the level goals to build the relaxed plan are $G_0, G_1, \ldots, G_{final\_layer}$, the set of helpful actions, $H(s)$, in state s are:

$$H(s) = \{a \in A \mid \text{eff}^+(a) \cap G_1 \neq \emptyset\}$$

For a complete explanation of the EHC algorithm see (Hoffmann & Nebel 2001).

The heuristic estimation applied by Metric-FF, as it only uses this scheme in problems without an optimization metric, is the number of actions in the relaxed plan (h_{ops}). We think that the same search scheme could be used in numerical problems with an optimization criteria using the heuristic measure proposed by this paper (h_{cost}). The justification is that using a weighted A* algorithm is too expensive in time and memory, and as will be seen in the experiments section, only few problems of some competition domains can be solved. For this reason, we have adapted the EHC algorithm to work with metrics. The resulting algorithm is shown in Figure 3. The only difference between this algorithm and the original one is that the latter performs breadth first search until a state s' with $h_{ops}(s') < h_{ops}(s)$ is found, while we introduce here a condition combining our heuristic, h_{cost}, with the previous one, h_{ops}. This combination assumes that when the cost is the same, shorter solutions regarding the number of actions are preferred (even when this was not included in the metric expression). Initially, we tried just replacing h_{ops} by h_{cost}, but the results were not good enough

Regarding the helpful actions we follow the same philosophy than Metric-FF, but the set of helpful actions is ordered by increasing costs. Thus, the first helpful action chosen is always the cheapest one. In our framework, to compute the set of helpful actions in the proposed algorithm for planning graph expansion, we have to take into account that actions delayed in the first level of the graph can achieve goals needed to build the relaxed plan in some level goal different of G_1 . Thus, the new set of helpful actions is defined as:

$$H(s) = \{a \in A \mid (\text{eff}^+(a) \cap G_1 \neq \emptyset) \vee (\exists i, \text{eff}^+(a) \cap GD_i \neq \emptyset)\}$$

where

$GD_i \subset G_i$ such that $\forall g \in GD_i, g \in \text{eff}^+(a')$ and $a' \in D_0$

```
initialize the current plan to the empty plan
$s := I$
while $h(s) \neq 0$ do
    starting from s, perform breadth first search for a state s'
    with $h_{cost}(s') < h_{cost}(s)$ or
    $(h_{cost}(s') = h_{cost}(s)$ and $h_{ops}(s') < h_{ops}(s))$
    avoiding repeated states using a hash table,
    not expanding states $s''$ where $h_{cost}(s'') = \infty$
    if no such state can be found then fail endif
    add the actions on the path to $s'$ at the end of the current plan
    $s := s'$
endwhile
output current plan, succeed
```

Figure 3: Adapted EHC algorithm for metric optimization problems.

In others words, an action is considered helpful if it achieves at least one of the lowest level goals in the relaxed plan, or it achieves at least one goal in some level i of the subset of goals of this level (represented as GD_i) achieved by actions delayed in the first level.

Experimental results

We have modified Metric-FF to behave according to out proposal. In this paper we only compare with the Metric-FF planner. It would be interesting to compare with other numerical planners, but the best way to see the performance of our approach is to change only the plan graph expansion algorithm and not other characteristics of the planner. We have performed the following experiments:

- Experiments using A* search. By means of these experiments we compare the performance between our proposed heuristic and the one used by Metric-FF.
- Experiments using the EHC algorithm, where we compare between the performance of Metric-FF and our implementation of EHC for problems with optimization criteria using the new heuristic.

The domains and problems used in the comparison for both types of experiments are some numeric domains from the 3rd International Planning Competition[1]. We ran the experiments on a Linux machine with 500 M Byte of memory running at 2 GHz. The maximum time allowed for the planner to find a solution was set to 300 seconds.

It is rather difficult to evaluate the performance of non-optimal planners when trying to achieve *good* solutions in terms of a quality metric, given that they have not guarantee on the optimality of the returned solutions. For this reason, to get a better understanding, we report the results showing the number of problems in which our approach performs strictly better, and separately the number of problems in which our approach performs better or equal than the Metric-FF planner.

Results using A*

Table 1 shows the number of problems solved in various domains, *Zenotravel*, *Driverlog*, *Satellite* and *Depots*, using the A* search algorithm. The number aside each domain name refers to the number of test problems.

[1]http://planning.cis.strath.ac.uk/competition/

Table 1: Problems solved with A*

Domain	Metric-FF	v1	v2	v3
Zenotravel(20)	11	7	7	7
Driverlog(20)	2	3	5	4
Satellite(20)	1	1	2	2
Depots(22)	6	3	8	7

As it can be observed, the A* algorithm fails on a large number of problems. In most cases, the planner failed to find a solution within the time bound. Also, v2 and v3 solve more problems than Metric-FF in three domains, though the difference is not very significant.

Tables 2 and 3 show the accumulated plan cost according to the problem metric defined, and the accumulated running times for *Zenotravel* and *Driverlog* domains respectively using A*. The first column refers to problems solved by all planners. The next columns refer to planners Metric-FF, and the three versions explained in previous section of our proposed algorithm to estimate the heuristic values. The results in the *Satellite* domain are not reported because the number of problems solved by any planner is very small. The results in the *Depots* domain are not reported because they performed exactly the same in quality and very similar in terms of time. This is due to the fact that in almost all the problems solved, the quality metric defined is plan length.

Table 2: Cost/Time in the Zenotravel domain using A*

Problem	Metric-FF	v1	v2	v3
pfile1	13564/0.01	13564/0.01	13564/0.00	13564/0.01
pfile2	6785/0.01	6785/0.01	6785/0.01	6785/0.01
pfile3	4507/0.02	4507/0.01	4507/0.01	4507/0.02
pfile4	20534/6.06	16972/3.22	16972/3.54	16972/0.53
pfile5	7424/0.07	3978/0.13	3978/0.16	3978/0.22
pfile6	20601/0.12	15434/0.23	15434/0.21	15434/0.28
pfile7	9538/4.7	8287/9.23	8287/10.4	8287/9.67
TOTAL	82953/10.99	69527/12.84	69527/14.33	69527/10.73

Table 3: Cost/Time in the Driverlog domain using A*

Problem	Metric-FF	v1	v2	v3
pfile1	1103/0.01	777/0.01	777/0.01	777/0.01
pfile3	883/254.57	657/102.66	657/104.33	657/45.64
TOTAL	1986/254.58	1434/102.67	1434/104.34	1434/45.65

In both domains, the costs obtained to v1, v2 and v3 is the same, and always better or equal the one obtained by Metric-FF. In the *Zenotravel*, the total quantitative gain is 16.19%, while all versions obtain strictly better results in 57.14% of solved problems. In terms of time, v1, v2 and v3 were 1.17 times slower, 1.30 times slower and 1.02 times

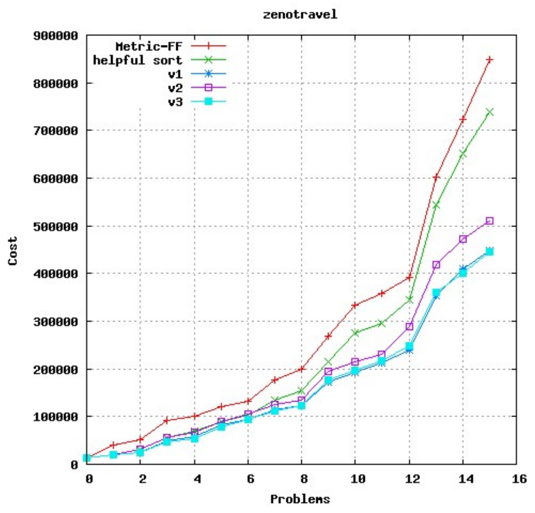

Figure 4: Cost in the Zenotravel domain using EHC.

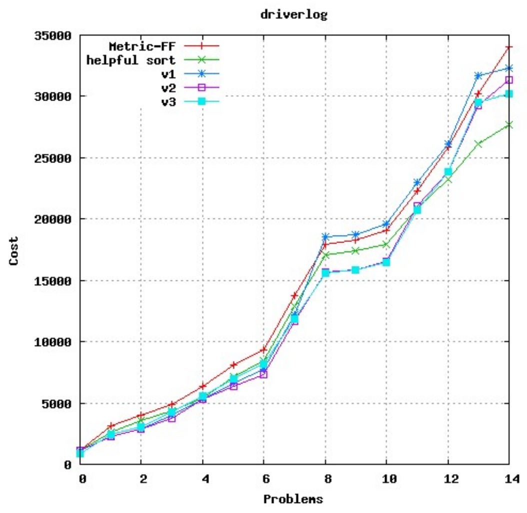

Figure 6: Cost in the Driverlog domain using EHC.

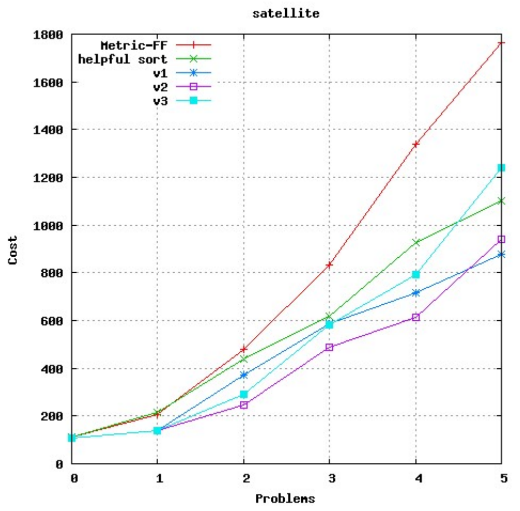

Figure 5: Cost in the Satellite domain using EHC.

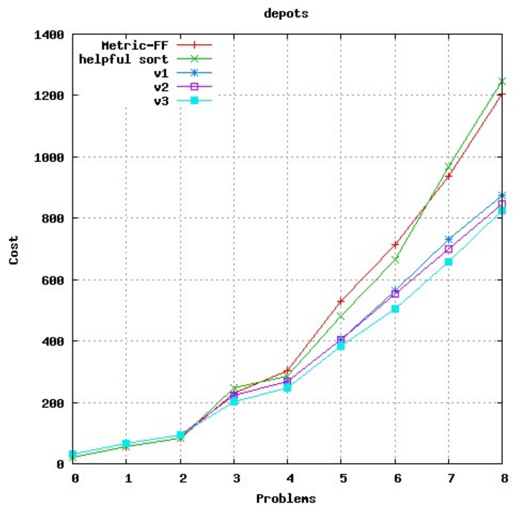

Figure 7: Cost in the Depots domain using EHC.

faster respectively. In the *Driverlog* domain, where only two problems were solved, the total quantitative gain is 27.79%. In both problems, the three versions perform strictly better than Metric-FF. In terms of time, v1, v2 and v3 were 2.48, 2.44 and 5.58 times faster respectively.

Results using EHC

Table 4 shows the number of problems solved in each case using the EHC search scheme. The second column shows the results obtained with Metric-FF applying EHC without having into account any optimization criteria, the third column refers to the same configuration just sorting the helpful actions, and the next three columns refer to the three versions of our approach using the EHC algorithm explained in this paper and sorting the helpful actions in increasing order of costs.

Table 4: Problems solved with EHC

Domain	Metric-FF	helpful-sort	v1	v2	v3
Zenotravel(20)	20	20	20	16	20
Driverlog(20)	16	16	15	16	15
Satellite(20)	13	16	9	13	9
Depots(22)	19	19	10	20	20

Regarding the cost of the obtained plans Figures 4, 6, 5 and 7 show the accumulated cost of the problems solved by all configuration in the aforementioned domains. As can be observed, all of our proposed approaches v1, v2 and v3, outperform Metric-FF in terms of quality metric. In fact, the heuristic obtained just sorting the helpful actions improve plan costs in almost all the domains except in the *Depots*. This can be well observed in Table 5 that sumarizes the quantitative gain in terms of cost in each case. The best improvements are achieved by v3 in the *Zenotravel* domain, *helpful-sort* in the *Driverlog* domain, *v1* in the *Satellite* domain, and v3 in *Depots* domain. Table 6 shows the percentage of problems with better and, in the row below, with better or equal performance comparing each approach first with Metric-FF (M-FF column in the table) and then (the *both* column) with both Metric-FF and the Metric-FF version just sorting the helpful actions (hs in the table). As it can be observed, the percentage of problems solved with smaller or equal cost than both is at least 53.3%.

Table 5: Total quantitative gain in terms of total cost.

Domain	helpful-sort	v1	v2	v3
Zenotravel	13.05%	47.35%	40.01%	47.68%
Driverlog	18.65%	5.18%	8.14%	11.31%
Satellite	34.86%	53.52%	44.11%	28.18%
Depots	-3.65%	27.33%	29.82%	31.40%

Table 6: Percentage of problems with better, and with better or equal performance in terms of the quality metric.

Domain	hs	v1		v2		v3	
	M-FF	M-FF	both	M-FF	both	M-FF	both
Zenotravel	55	75	55	62.5	50	85	65
	65	80	60	75	62.5	90	70
Driverlog	56.2	53.3	40	68.7	50	80	53.3
	87.5	66.6	53.3	81.2	62.5	80	53.3
Satellite	75	100	62.5	100	63.6	88.8	55.5
	91.6	100	62.5	100	63.6	88.8	55.5
Depots	10.5	77.7	66.6	42.1	36.8	42.1	36.8
	84.2	100	88.8	100	94.7	100	94.7

These results support our initial intuition about the accuracy of non-admissible heuristics (even using a Hill-Climbing algorithm). In most cases the quality of the plans is improved because the heuristic estimation used, though non-admissible, is closer to the real optimal value.

We do not report in this paper all the results with respect the average number of operators in solution plans, but in the *Zenotravel*, the *Driverlog* and the *Satellite* domains, the plans obtained using our heuristics are in general larger than the obtained by Metric-FF. As we have commented previously, this is a consequence of having a more accurate heuristic in terms of cost because a plan with a small number of operators can suppose a great cost with respect the quality metric and viceversa. For instance, in the *Zenotravel* domain, the average number of operators per plan is 26.8 for Metric-FF and 26.7 for helpful-sort, while for the versions using our heuristics are 34.2 for v1, 34.3 for v2 and 37 for v3.

One would expect that improving the quality of the resulting plans would get worse the time to achieve these plans. It seems reasonable because to find a better solution means usually more searching, and therefore to build more relaxed planning graphs, thus spending more time. However, the results in the *Depots* domain show that this is not always true. Figures 8, 10, 9 and 11 show the accumulated CPU time in seconds.

The results regarding *Zenotravel*, *Driverlog* and *Satellite* domains show that the differences in time are not relevant for easy problems, but when the problems are harder (this happens with the last problems of each domain in the planning competition) they are rather significant. For example, in the *Zenotravel* domain the differences until problem 12 are not relevant, as the time used by all planners is smaller than one second. But due to the last 3 problems, v3 was 4.49 times slower, as Table 7 shows. The same happens in *Driverlog* using v3 with problem 13. In this case, the time consumed was especially high. The worst time was obtained for v1 in *Satellite*. In all cases in which the time is worse than Metric-FF, the quality of solutions is better. However, in the *Depots* domain the heuristics proposed in this paper perform especially well in terms of time. In this case, problems were solved 21.38 and 17.26 times faster than Metric-FF, using v1 and v3 respectively. We have not yet clear explanation for this phenomenon, but the quality metric of all the problems which contribute more to the differences is defined as *fuel-used* and not in terms of plan length. On the other hand, for all these problems, all the other parameters that could be considered (as search time, number of explored states, time of building the relaxed planning graphs, and even number of actions in solution plans) are quite smaller than for Metric-FF. Furthermore, the quality of the plans obtained for these problems was improved.

Table 7: Time summary

Domain	helpful-sort	v1	v2	v3
Zenotravel	1.11 slower	2.41 slower	3.11 slower	4.49 slower
Driverlog	1.21 faster	0.67 faster	0.87 faster	9.04 slower
Satellite	0.26 faster	27.23 slower	8.81 slower	21.60 slower
Depots	1.59 slower	21.38 faster	7.18 faster	17.26 faster

Conclusions and future work

This paper presents how an existing method to compute heuristics in heuristic planning, the relaxed planning graph, can be improved to deal with problems where a metric optimization criteria is defined. The aim was to achieve *good*, though provably suboptimal, plans regarding this metric. The possible definition of a metric optimization criteria is a feature of numeric domains. In this paper we do not deal with the other features of these domains, but we have implemented our algorithm on the Metric-FF planner, which implements methods to deal with it.

The proposed improvements allow to compute more accurate, though non-admissible, heuristics. As traditional relaxed planning graph, the algorithm we propose is independent of the planner as well as the domain used and the

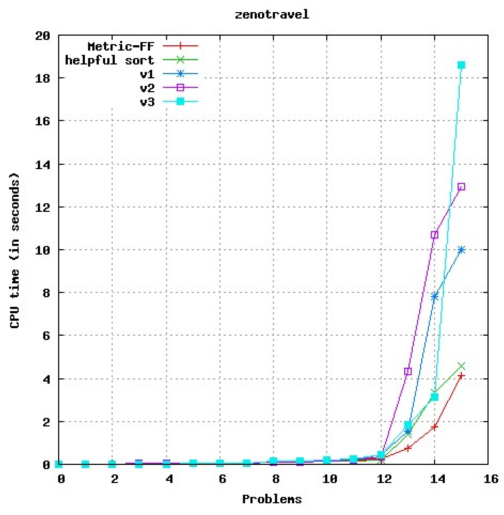

Figure 8: Time in the Zenotravel domain using EHC.

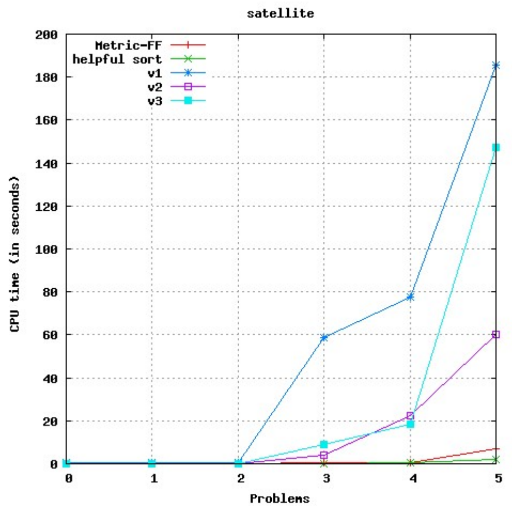

Figure 9: Time in the Satellite domain using EHC.

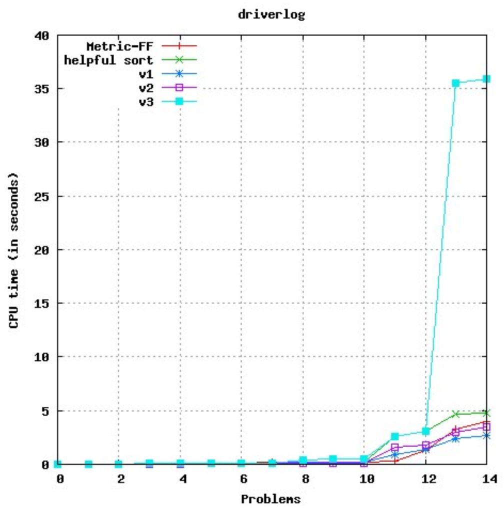

Figure 10: Time in the Driverlog domain using EHC.

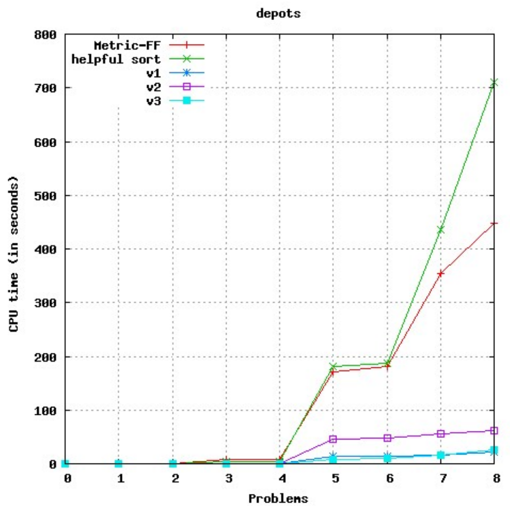

Figure 11: Time in the Depots domain using EHC.

problem to solve. Therefore, other heuristic planners with a management of characteristics of numerical domains could benefit of it.

Additionally, we propose an adaptation of the Hill-Climbing algorithm used by Metric-FF (called Enforced Hill Climbing) that can use heuristics representing costs relative to a quality metric. Usually, heuristic planners whose objective is not only to find plans but good ones, use some kind of A* search algorithm. This is the case of Metric-FF.

We performed experiments using both, an A* algorithm and the proposed Hill-Climbing algorithm. Results show that the quality of resulting plans is improved always with the A* algorithm and in a high percentage of cases with EHC. The number of solved problems is always bigger using the latter as the comsuption of resources (especially time) is smaller. Therefore, we think this is a good alternative even when the objective of planning is to optimize a metric criteria.

As future work we think it would be interesting to perform experiments using the weighted A* algorithm with different values of w, or use the proposed heuristics in different planners.

In this paper we focused mainly in comparisons with the Metric-FF planner, but it would be interesting also to study the behaviour of the three non-admissible heuristics we propose more deeply. This could help to explain better some of the results presented.

References

Blum, A. L., and Furst, M. L. 1995. Fast planning through planning graph analysis. In Mellish, C. S., ed., *Proceedings of the 14th International Joint Conference on Artificial Intelligence, IJCAI-95*, volume 2, 1636–1642. Montréal, Canada: Morgan Kaufmann.

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129(1-2):5–33.

Do, M. B., and Kambhampati, S. 2003. Sapa: A scalable multi-objective heuristic metric temporal planner. *Journal of Artificial Intelligence Research* 20:155–194.

Fox, M., and Long, D. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research* 20:61–124.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.

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

Long, D., and Fox, M. 2003. The 3rd international planning competition: Results and analysis. *Journal of Artificial Intelligence Research* 20:1–59.

McDertmott, D. 1996. A heuristic estimator for means-ends analysis in planning. In *Proceedings of the 3rd International Conference on Artificial Intelligence Planning Systems (AIPS-96)*, 142–149. AAAI Press.

Refanidis, I., and Vlahavas, I. 2001. The GRT planning system: Backward heuristic construction in forward state-space planning. *Journal of Artificial Intelligence Research* 15:115–161.

Sapena, O., and Onaindía, E. 2004. Handling numeric criteria in relaxed planning graphs. In *Advances in Artificial Intelligence. IBERAMIA, LNAI 3315*, 114–123. Springer Verlag.