

Logical Encodings With No Time Indexes for Defining and Computing Admissible Heuristics for Planning

Miquel Ramirez

Universitat Pompeu Fabra
Passeig de Circumvalació 8
08003 Barcelona Spain
miquel.ramirez@upf.edu

Blai Bonet

Departamento de Computación
Universidad Simón Bolívar
Caracas 1080-A, Venezuela
bonet@ldc.usb.ve

Héctor Geffner

ICREA & Universitat Pompeu Fabra
Passeig de Circumvalació 8
08003 Barcelona Spain
hector.geffner@upf.edu

Abstract

A limitation of the SAT approach to planning and the more recent Weighted-SAT approach to planning with preferences is the use of logical encodings where every fluent and action must be tagged with a time index. The result is that the complexity of the encodings grows exponentially with the planning horizon, and for metrics other than makespan, the optimality achieved is conditional on the planning horizon used. In this work, we consider the use of logical encodings in planning but for defining and computing *admissible heuristics* only, for which no time indices or planning horizons are required. The basic logical formulation, following a recent proposal by Bonet and Geffner in KR-06, captures a generalization of the optimal delete-relaxation heuristic, which is then extended with *implicit plan constraints* for boosting their values by capturing information lost in the delete-relaxation, and with a structural relaxation scheme for CNF formulae recently proposed by Ramirez and Geffner in CP-07 that reduces the *treewidth* of the theory to any bound w , producing thus *poly-time admissible heuristics* h^w that are able to handle costs and rewards, and (some) delete information. The experimental results, although preliminary, show that in some domains, these heuristics are cost-effective and can be competitive with the best known heuristics.

Introduction

Propositional encodings form the basis of the SAT approach to planning where planning problems are translated into CNF formulae that are fed into state-of-the-art SAT solvers (Kautz & Selman 1996). This approach has been recently extended to deal with planning with preferences where the use of SAT solvers is replaced with MAX-Weighted SAT or Weighted CSP solvers (Brafman & Chernyavsky 2005; E. Giunchiglia 2007). While progress in SAT and Weighted-SAT solver technology has been significant in recent years, a limitation of these approaches is that the complexity of the encodings grows exponentially with the planning horizon, and that for metrics other than makespan, the optimality of the resulting plans is conditional on the planning horizon used.

In this work, we build on a recent proposal for using logical encodings with no time indices for defining and computing admissible heuristics (Bonet & Geffner 2006), so that the

heuristic of a state s is given by the cost of the best model of the theory conditioned on s . The basic heuristic corresponds to a generalization of the optimal delete-relaxation heuristic capable of dealing with rewards as well. We then consider two extensions of this idea: the use of valid *plan constraints* to boost the heuristic by capturing information lost in the delete-relaxation, and the use of a relaxation scheme, proposed recently in (Ramirez & Geffner 2007), that reduces the theory *treewidth* to any bound w , producing *poly-time admissible heuristics* h^w able to handle both costs and rewards, and (some) delete information, closely related to the *mini-buckets heuristics* (Dechter & Rish 2002). The experimental results, although preliminary, show that in some domains, these heuristics are cost-effective and can be competitive with the best known heuristics like h^2 (Haslum & Geffner 2000).

Planning Model

Following (Bonet & Geffner 2006), abbreviated BG from now on, we deal with planning problems where actions have non-negative costs and fluents have costs that can be positive, negative, or zero. Fluents with positive costs are called *penalties*, and those with negative costs *rewards*. For simplicity, we deal only with Strips operators and no conditional effects.

A planning problem is thus a tuple $P = \langle F, I, O, G \rangle$ where F is the set of relevant atoms or fluents, $I \subseteq F$ and $G \subseteq F$ are the initial and goal situations, and O is a set of grounded actions a with preconditions, adds, and deletes $Pre(a)$, $Add(a)$, and $Del(a)$, all subsets of F .

A plan π is an applicable action sequence a_0, a_1, \dots, a_n with $a_i \in O$ for $i = 0, \dots, n$, that transforms the initial state s_0 associated with I into a final state s_{n+1} where the goal G holds.

In this work, we are interested in plans π that minimize the cost $c(\pi)$, defined as

$$c(\pi) \stackrel{\text{def}}{=} \sum_{a_i \in \pi} c(a_i) + \sum_{p \in F(\pi)} c(p) \quad (1)$$

where $F(\pi)$ is the set of fluents made true by plan π at any time point during the execution, while $c(a_i)$ and $c(p)$ stand for the cost of the action a_i and the cost of the fluent p respectively. Action costs $c(a)$ are assumed to be *non-*

negative, while fluent costs $c(p)$ can be *positive* (penalties), *negative* (rewards), or *zero* (assumed by default).

We denote by $c^*(P)$ the cost of the best plan for P

$$c^*(P) \stackrel{\text{def}}{=} \min\{c(\pi) : \pi \text{ is a plan for } P\} \quad (2)$$

and set $c^*(P)$ to ∞ when P admits no plan.

We call the planning cost model captured by (1) the *penalties and rewards* model, abbreviated as PR. This model is similar to the one used in over-subscription or partial-goal planning where due to constraints or preferences, it may not be possible or convenient to achieve all the goals (Smith 2004; Van den Briel *et al.* 2004). The main difference is that that atoms in PR can be rewarded when they are achieved *anytime* during the execution of the plan, not only when they are achieved at the *end*. Yet, if the atoms p with positive reward (i.e., negative costs) are *persistent* (cannot be deleted), the two models are the same, otherwise soft goals can be expressed as rewards by a simple transformation described in BG. The model is not fully general but is general enough for our purposes, as it extends the cost model for classical planning significantly, and in particular, ‘hard goals’ are not needed as they can be expressed as highly rewarded ‘soft goals’.

Heuristic h_c^+

The heuristic formulated by BG for the PR model builds on the optimal delete-relaxation heuristic proposed in classical planning. If P^+ is the delete-relaxation of problem P , i.e., the planning problem obtained by dropping the deletes $Del(a)$, the heuristic $h_c^+(P)$ that provides an estimate of the cost of solving P given the cost function c is defined as

$$h_c^+(P) \stackrel{\text{def}}{=} c^*(P^+). \quad (3)$$

where $c^*(P^+)$ is the optimal cost of the delete-relaxation. For the 0/1 cost function that characterizes classical planning, where the cost of all fluents is 0 and the cost of all actions is 1, this definition yields the (optimal) delete-relaxation heuristic which provides an estimate of the number of steps to the goal. Expression (3) generalizes this heuristic to the larger class of cost functions where actions may have non-uniform costs and atoms can be rewarded or penalized. In all cases, the heuristics is *admissible*:¹

Proposition 1 (Admissibility) $h_c^+(P) \leq c^*(P)$.

If $P[I = s]$ and $P[G = g]$ refer to the planning problems that are like P but with initial situation $I = s$ and goal situation $G = g$ respectively, then (optimal) forward heuristic-search planners aimed at solving P need to compute the heuristic values $h_c^+(P[I = s])$ for *all* states s encountered, while regression planners need to compute the heuristic values $h_c^+(P[G = g])$ for *all* encountered subgoals g . Since each such computation is intractable, even for the 0/1 cost function, classical planners like HSP and FF settle on polynomial but non-admissible approximations. In BG, a different approach is taken: rather than performing an intractable computation for *every search state* encountered, an

¹In the presence of conditional effects, additional requirements are needed for admissibility; see BG.

intractable computation is performed *once* by compiling a suitable propositional theory. We review this next and then consider other uses of this theory.

Encoding the Heuristics in Logic

MinCostSAT is a variant of SAT where the *best satisfying truth assignment* over a given CNF formula is sought with *the cost of an assignment given by the sum of the costs of the literals it makes true* (Li 2004; Fu & Malik 2006).

MinCostSAT is closely related to Weighted-MAX SAT (Larrosa & Heras 2005; Li, Manyá, & Planes 2006), where the cost of an assignment is given by the sum of the costs of the clauses it violates, and it is rather simple to express instances of one in terms of the other. Both formulations have many applications, and as the last Weighted-MAX SAT competition shows,² there are many good MinCostSAT and Weighted-MAX SAT solvers, most of which are based on DPLL-style search. MinCostSAT can also be solved without search by compiling the theory into d-DNNF, and moreover once a theory T is compiled into d-DNNF, *all* the MinCostSAT problems $T \cup S$ for *any* set of literals S in T can be computed in linear-time from the compiled representation (Darwiche & Marquis 2004). Bonet and Geffner take advantage of this property for computing *all* the heuristic values $h_c^+(P[I = s, G = g])$ in time linear in the size of the compilation of a logical encoding $L(P)$ of the planning problem P with the initial and goal situations removed.

The logical encoding $L(P)$ avoids the use of time indices and is built using ideas from Logic Programming (Lloyd 1987), as the logic program made up of the rules

$$p \leftarrow Pre(a), a \quad (4)$$

for each (positive) effect p of an action a with preconditions $Pre(a)$ in P , and rules

$$p \leftarrow set(p). \quad (5)$$

for each fluent p , where $set(p)$ is ‘dummy’ action for ‘setting’ the initial situation to s when computing the heuristic values $h_c^+(P[I = s])$ in a progression search. No such encoding trick is needed to support a regression search.

The logic program $L(P)$ is mapped into a propositional formula $wffc(L(P))$ whose models encode the intended models of the program: the models in which each fluent p has a *well-founded support* in terms of actions that are true in the model. This completion can be obtained from $L(P)$ by a polynomial transformation along the lines of (Lin & Zhao 2003).

If $I(s)$ stands for the collection of unit clauses $set(p)$ that represent a state s , namely $set(p) \in I(s)$ iff $p \in s$, and $\neg set(p) \in I(s)$ iff $p \notin s$, the correspondence between *heuristics* and *the cost of the best models of a theory T* , called the cost or rank of the theory and denoted as $r^*(T)$, can be expressed as:

Proposition 2 For any initial situation s , goal g , and cost functions c ,

$$h_c^+(P[I = s, G = g]) = r^*(wffc(L(P)) \cup I(s) \cup g)$$

²<http://www.maxsat07.udl.es>.

where the cost $r(l)$ of positive literals l is $c(l)$, and the cost of negative literals is 0.

From this result and the properties of d-DNNF formula, Bonet and Geffner obtain:

Theorem 3 *Let $\Pi(P)$ refer to the compilation of the theory $\text{wffc}(L(P))$ into d-DNNF. Then for any initial and goal situations s and g , and any cost function c , the heuristic value $h_c^+(P[I = s, G = g])$ can be computed from $\Pi(P)$ in linear time.*

Since the cost $r^*(T)$ of a theory in d-DNNF can be computed by a circuit obtained from T by replacing ANDs by Sums and ORs by Mins (Darwiche & Marquis 2004), Proposition 2 and Theorem 3 can be understood as asserting that the heuristic values $h_c^+(P[I = s, G = g])$ for any s and g can be computed by means of a circuit obtained from the compiled formula $\Pi(P)$, whose input are the initial and goal situations $I = s$ and $G = g$ and whose output is the heuristic value $h_c^+(P[I = s, G = g])$.

Boosting the Heuristic: Plan Constraints

The logical formulation of the heuristic h_c^+ suggests a number of improvements. We consider first an extension that results from taking *valid plan constraints* into account; these are constraints satisfied by some optimal plan (if there is a plan at all) that have no effect on the *optimal cost of a problem* but can boost the *heuristic function* by capturing information that is lost in the delete-relaxation (Bonet & Geffner 2007).

A plan constraint C is a propositional formula over the sets of actions and fluents A and F . A plan π for a problem P satisfies a constraint C , written $\pi \models C$, if C is true over the interpretation that makes true only the actions and fluents in π and $F(\pi)$. Thus, a plan constraint $p \vee q$ when p and q are fluents is satisfied by π when π makes p or q true at some point in the execution, while π satisfies $\neg p \vee \neg q$ when at least one the two fluents p or q is never made true by π . Notice that a mutex (p, q) says something else; namely, that no reachable state can make both p and q at the same time, and hence does not express necessarily a valid plan constraint. On the other hand, the constraint that prevents two moves away from one vertex in a TSP, is an example of a valid plan constraint. Plan constraints as defined above are not as expressive as modal or temporal formulae, yet they are simple and sufficient for illustrating how they can be used to improve the delete-relaxation heuristic once the heuristic is formulated in logic.

Plan constraints are not used to modify the definition of plans but rather their *costs*, so that plans that do not comply with the constraints get an infinite cost. The *constrained cost* of a plan is thus defined as

$$c(\pi, C) \stackrel{\text{def}}{=} \begin{cases} c(\pi) & \text{if } \pi \models C \\ \infty & \text{otherwise} \end{cases} \quad (6)$$

where the *constrained optimal cost* of a problem as

$$c^*(P, C) \stackrel{\text{def}}{=} \min_{\pi} c(\pi, C) \quad (7)$$

with π ranging over the plans for P . Clearly if C is a valid constraint for problem P , it cannot change the cost of P ,

and hence $c^*(P, C) = c^*(P)$. When C is valid in P but not valid in the delete-relaxation P^+ , however, C can boost the value of the heuristic defined as

$$h_c^+(P, C) \stackrel{\text{def}}{=} c^*(P^+, C) \quad (8)$$

Theorem 4 (Admissibility and Boosting) *Let C be valid plan constraint for P under the cost function c . Then*

$$h_c^+(P) \leq h_c^+(P, C) \leq c^*(P, C) = c^*(P),$$

where both inequalities can be strict.

This is important as it says that the value of the delete-relaxation heuristic h_c^+ can be increased, while preserving admissibility, by making explicit certain valid but implicit plan constraints. The logical account of this new heuristic is a direct extension of Proposition 2 above:

Proposition 5 (Constrained Heuristic and Ranks) *For any initial situation s , goal g , cost function c , and plan constraint C ,*

$$h_c^+(P[I = s, G = g], C) = r^*(\text{wffc}(L(P)) \cup C \cup I(s) \cup g)$$

for r such that $r(l) = c(l)$ for positive literals l and $r(l) = 0$ otherwise.

From this result and the properties of formulae in d-DNNF, once again we obtain:

Theorem 6 (Computing Constrained Heuristics)

Let $\Pi(P, C)$ refer to the compilation of the theory $\text{wffc}(L(P)) \cup C$ into d-DNNF. Then for any initial and goal situations s and g , and any cost function c , the heuristic value $h_c^+(P[I = s, G = g], C)$ can be computed from $\Pi(P, C)$ in linear time.

Once the heuristics are formulated in logic, they can be computed using MinCostSAT or Weighted-Max SAT solvers. The appeal of the compilation-based approach captured in Theorems 3 and 6 is that it yields a circuit that map situations into heuristic values in linear time. On the other hand, the size of these circuits can be exponential in the number of variables, a problem that we address next.

Heuristics and Treewidth Relaxation

The compilation of a CNF formula into d-DNNF, as many operations over graphical models (CNFs, CSPs, Bayesian Networks, etc.) is exponential in the model *treewidth* (Darwiche 2001), a parameter that measures the degree of interaction among the variables in the model as captured in the underlying interaction graph (Dechter 2003).

Recently, Ramirez and Geffner have introduced a scheme for relaxing graphical models, and in particular SAT encodings, so that their induced treewidth can be bound by a given parameter w (Ramirez & Geffner 2007).³ This is accomplished in a simple way by *mapping different occurrences of the same variable x_i in T , into different (new) variables x_i^j* . By choosing the variables x_i to rename along with their new

³Actually, the scheme reduces the *hypertree width* of the theory which unlike the standard treewidth is not affected by the arity of the clauses (Gottlob, Leone, & Scarcello 2001). Here we will ignore this difference.

names, a relaxed theory T^- can be obtained with treewidth bounded by any given w . This can be done for example, by selecting a w -cutset of T , and then ensuring that each occurrence of each variable in the w -cutset, gets a different name in T^- . A w -cutset is a set of variables in T whose instantiation ensures that the resulting theory has treewidth bounded by w (Dechter 1990).

This relaxation scheme was used in (Ramirez & Geffner 2007) for developing a MinCostSAT solver that searches for a min-cost assignment for T using lower bounds obtained from the d-DNNF compilation of the relaxation T^- . Here we use it for defining a family of heuristics h_c^w exponential in the treewidth bound w . For this, we replace the theory $T = \text{wffc}(L(P)) \cup C$ in Proposition 5 by its relaxation T^- :

Definition 7 (Heuristics h_c^w) For any initial situation s , goal g , cost function c , and plan constraint C ,

$$h_c^w(P[I = s, G = g], C) \stackrel{\text{def}}{=} r^*(T^- \cup I(s) \cup g)$$

where T^- is the theory obtained from $T = \text{wffc}(L(P)) \cup C$ by renaming fully the variables in a w -cutset of T .

In analogy to Theorem 6, the heuristic values $h_c^w(P[I = s, G = g], C)$ can be computed for any s , g , and c in linear time from the compilation of the relaxed theory T^- , an operation that is now exponential in the target treewidth w that we control.

For any fixed w , h_c^w is a *poly-time and admissible heuristic capable of handling both costs and rewards, using the delete information that is captured in the planning constraints C* . It is an heuristic different from the existing ones that explicitly builds on the theory of graphical models, with close connections to the *mini-buckets heuristics* used in constrained optimization problems (Dechter & Rish 2002).

Of course, there is a trade off: as w increases the heuristic h_c^w approaches the value of the heuristic h_c^+ (with or without constraints) but its compilation becomes more difficult and the computation of the heuristic values becomes slower, as the compiled d-DNNF representation gets larger as well.

Preliminary Experimental Results

Experimental results on the use of the generalized delete-relaxation heuristic h_c^+ for optimal planning with costs and rewards are reported in (Bonet & Geffner 2006). Here, we add a few more results, considering the use of plan constraints and the treewidth relaxation. The search algorithm is a slight variant of the A* algorithm that ensures optimal solutions even in the presence of *negative costs* when the heuristic is *monotonic* (all the heuristics above are monotonic; implying that the evaluation function $f(n) = g(n) + h(n)$ does not decrease along any path in the search). The search is backwards from the goal, and a table of structural mutexes is built prior to the search, so that states reached in the regression that contain a mutex, and hence are not reachable from the initial state, are pruned (Bonet & Geffner 2001).

Two domains are considered: Serialized Logistics, a serialized version of the Logistic instances in the Second IPC (Bacchus 2001) and Rock Analyst, a version of the domain discussed in (Smith 2004) where rocks of different classes

must be picked and analyzed from various locations. The data is reported in Tables 1 and 2, showing the compilation time (C), the search time (S), and the number of expanded nodes (N). In the first domain, results for the heuristics h_c^+ and h_c^w are reported, with w set in each case to the value that minimizes overall time (search plus compilation). Selecting the value of w automatically remains a topic for future research. In the second domain, the two heuristics are reported in two forms: with a valid plan constraint and without it (constrained and unconstrained). In the two tables, the data for the heuristic h^2 (Haslum & Geffner 2000) is also included as a reference, reporting the time to precompute this heuristic in the H column.

We can see in Table 1 that the heuristics h_c^+ and h_c^w for suitable values of w , scale up better than h^2 in the Serialized Logistics domain. The heuristic h^2 can be computed faster than either h_c^+ or h_c^w but is less informed. E.g., in the largest problem solved using h^2 , the search expands 99827 nodes, while h_c^+ and h_c^w expand 727 and 2129 nodes respectively. The heuristic h_c^w is weaker than h_c^+ but can be computed faster. As a result, in the largest 4 instances, it expands 4-6 times more nodes but the search takes 4-5 times less.

Rock Analyst is a variation of the TSP in which there are a number of rocks of different classes at various locations and the goal is to analyze a rock from each class. The optimal plan thus includes a minimum cost path that travels along the locations that contain rocks of all classes but without having to visit all locations. The problem combines two tasks that are both intractable: the selection of the rock in each class to visit (Set Cover), and the selection of the tour to visit them (TSP). In the constrained heuristics, C is the valid plan constraint that rules out moving away from the same location twice (the constraint is valid given that the costs are Euclidean).

Table 2 shows the compilation and search results for problems with n locations and $3n$ rocks classified into n classes. In this case, the heuristics h_c^+ and h_c^w do much better with the constraint C than without, with the constrained h_c^w doing best for suitable values of the treewidth parameter w , and both heuristics expanding very few nodes. The heuristic h^2 expands a much larger number of nodes, but expands them much faster, and thus the overall search time is only slightly worse than h_c^w . The preprocessing for computing the heuristic h^2 is also expensive in this domain: as expensive as the compilation in the largest problems and more expensive than the compilation in the smallest ones. Yet, the time for computing h^2 grows polynomially, while the time of the compilation grows exponentially. The opposite happens on the other hand with the number of expanded nodes. So none of these heuristics end up solving the problem for $n = 10$.

Summary

The h_c^+ heuristic is a logical formulation of the delete-relaxation heuristic, defined in terms of logic programs and their well-founded completions, that uses no time-indexed propositions and handles both penalties and rewards. By compiling the resulting theory into d-DNNF, a circuit is obtained that maps situations into their heuristic values in linear-time. The semantic limitation of the formulation

	h^2 N/H/S	h_c^+ N/C/S	h_c^w w/N/C/S
4-0	4,295/0.7/0.2	40/0.2/0.0	16/1,135/0.2/0.1
4-1	7,079/0.7/0.3	109/0.2/0.1	16/2,455/0.2/0.2
4-2	537/0.6/0.0	25/0.2/0.0	16/424/0.2/0.0
5-0	118,389/0.6/5.7	490/0.2/0.5	16/16,407/0.2/1.4
5-1	7,904/0.6/0.4	103/0.2/0.1	16/2,123/0.2/0.2
5-2	143/0.6/0.0	8/0.2/0.0	16/33/0.2/0.0
6-0	316,175/0.6/18.1	668/0.2/0.7	16/14,718/0.2/1.3
6-1	1,489/0.7/0.1	19/0.2/0.0	16/413/0.2/0.0
6-2	301,054/0.7/17.6	517/0.2/0.4	14,654/0.2/1.3
6-3	99,827/0.7/5.6	727/0.2/0.5	16/2,129/0.2/0.2
7-0	—	4,973/1.1/58.6	24/57,392/1.4/32.6
7-1	—	175,886/1.1/2,648.4	25/4,456,121/1.4/904
8-0	—	591/1.0/5.4	24/13,126/1.4/6.4
8-1	—	12,913/1.1/191.6	24/427,943/1.5/978.7
9-0	—	3,083/1.0/42.0	24/60,834/1.5/43.2
9-1	—	81/1.1/0.6	24/4,056/1.4/1.8
10-0	—	—	32/463,490/7.3/1472.6
10-1	—	20,220/5.6/997.7	32/90,628/7.5/215.8
11-0	—	20,143/5.2/1,855.8	32/133,778/7.5/533.6
12-0	—	23,556/5.1/1,769.8	32/128,004/8.1/446.7

Table 1: Compilation and Search for Serialized Logistics over first 18 instances from 2nd IPC. N stands for the number of nodes expanded, H for the time spent computing the h^2 heuristic, C for the compilation time, and S for search time. w denotes the value of w for which the heuristic h_c^w showed best performance (min search + compilation time).

n	h^2 N/H/S	h_c^+ unconstr. N/C/S	h_c^+ constr. N/C/S	h_c^w unconstr. w/N/C/S	h_c^w constr. w/N/C/S
4	16/0.8/0.0	5/0.0/0.0	5/0.0/0.0	4/39/0.0/0.0	11/5/0.0/0.0
5	19/3.8/0.0	6/0.1/0.0	6/0.1/0.0	12/75/0.1/0.0	17/75/0.1/0.0
6	29/13.0/0.0	13/0.4/0.0	13/0.3/0.0	20/157/0.2/0.0	29/13/0.2/0.0
7	914/37.1/0.5	43/4.4/1.6	35/1.7/1.0	36/195/1.5/0.5	54/35/1.0/0.1
8	7,298/91.6/6.5	19/71.5/15.0	10/30.9/4.6	12/22,086/18.0/64.0	54/10/24.1/0.4
9	14,052/205.1/16.0	-/-/-	11/253.3/58.4	12/19,326/333.4/521.7	64/11/209.6/4.7

Table 2: Search and Compilation for Rock Analyst with n locations and $3n$ rocks classified into n classes. The heuristics h_c^+ and h_c^w used with and without the plan constraint C that prevents moving away twice from the same location (constrained and unconstrained). N stands for the number of nodes expanded, H for the time spent computing the h^2 heuristic, C for the compilation time, and S for search time. w denotes the value of w for which the heuristic h_c^w showed best performance (min search + compilation time).

in (Bonet & Geffner 2006) is that it is still based on the delete-relaxation, while its computational limitation is that the compilation can be exponential in the treewidth of the theory. In this work we have addressed these limitations by extending the formulation with valid plan constraints that capture information that is lost in the delete relaxation, and by introducing a family of weaker heuristics h_c^w , that for a fixed w can be computed by circuits of polynomial size. The experimental results, although preliminary, show that in some domains, these heuristics are cost-effective and can be competitive with the best known heuristics. Topics of future research include the automatic derivation of valid plan constraints, the automatic selection of the target treewidth parameter w , and the development of more effective relaxation schemes that maximize the value of the heuristics h_c^w for a given w .

Acknowledgements. H. Geffner is partially supported by Grant TIN2006-15387-C03-03 from MEC, Spain.

References

- Bacchus, F. 2001. The 2000 AI Planning Systems Competition. *Artificial Intelligence Magazine* 22(3).
- Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129(1–2):5–33.
- Bonet, B., and Geffner, H. 2006. Heuristics for planning with penalties and rewards compiled knowledge. In *Proc. KR-06*.
- Bonet, B., and Geffner, H. 2007. Heuristics for planning with penalties and rewards formulated in logic and computed through circuits. Technical report, UPF.
- Brafman, R. I., and Chernyavsky, Y. 2005. Planning with goal preferences and constraints. In *Proc. ICAPS-05*, 182–191.
- Darwiche, A., and Marquis, P. 2004. Compiling propositional weighted bases. *Artif. Intell.* 157(1-2):81–113.
- Darwiche, A. 2001. Decomposable negation normal form. *J. ACM* 48(4):608–647.
- Dechter, R., and Rish, I. 2002. Mini-buckets: A general scheme for approximating inference. *Journal of the ACM* 107–153.
- Dechter, R. 1990. Enhancement schemes for constraint processing: Backjumping, learning, and cutset decomposition. *Artificial Intelligence* 41(3):273–312.
- Dechter, R. 2003. *Constraint Processing*. Morgan Kaufmann.
- Fu, Z., and Malik, S. 2006. Solving the Minimum Cost Satisfiability Problem using SAT Based Branch-and-Bound Search. In *Proc. of ICCAD'06*.
- Giunchiglia, E., and Maratea, M. 2007. Planning as satisfiability with preferences. In *Proc. AAAI-07*. At <http://www.star.dist.unige.it/>
- Gottlob, G.; Leone, N.; and Scarcello, F. 2001. Hyper-tree decompositions: A survey. In Sgall, J.; Pultr, A.; and Kolman, P., eds., *Mathematical Foundations of Computer Science 2001, 26th International Symposium*, 37–57.
- Haslum, P., and Geffner, H. 2000. Admissible heuristics for optimal planning. In *Proc. of the Fifth International Conference on AI Planning Systems (AIPS-2000)*, 70–82.
- Kautz, H., and Selman, B. 1996. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of AAAI-96*, 1194–1201. AAAI Press / MIT Press.
- Larrosa, J., and Heras, F. 2005. Resolution in Max-SAT and its Relation to Local Consistency in Weighted CSPs. In *Proc. of IJCAI-05*, 193–199.
- Li, C. M.; Manyá, F.; and Planes, J. 2006. Detecting Disjoint Inconsistent Subformulas for Computing Lower Bounds for Max-SAT. In *Proc. AAAI'06*.
- Li, X. Y. 2004. *Optimization Algorithms for the Minimum-Cost Satisfiability Problem*. Ph.D. Dissertation, North Carolina State University.
- Lin, F., and Zhao, J. 2003. On tight logic programs and yet another translation from normal logic programs to propositional logic. In *Proc. IJCAI-03*, 853–858.
- Lloyd, J. 1987. *Foundations of Logic Programming*. Springer-Verlag.
- Ramirez, M., and Geffner, H. 2007. Structural relaxations by variable renaming and their compilation for solving mincostsat. In *Proc. CP-07*. To Appear.
- Smith, D. E. 2004. Choosing objectives in over-subscription planning. In *Proc. ICAPS-04*, 393–401.
- Van den Briel, M.; Nigenda, R. S.; Do, M. B.; and Kambhampati, S. 2004. Effective approaches for partial satisfaction (over-subscription) planning. In *Proc. AAAI 2004*, 562–569.