# HEURISTIC SEARCH PLANNER 2.0

*Blai Bonet and Hector Geffner*

■ We describe the HSP2.0 planning algorithm that entered the second planning contest held at the Fifth International Conference on Artificial Intelligence Planning and Scheduling (AIPS'00). HSP2.0 is a domain-independent planning algorithm that implements the family of heuristic search planners that are characterized by the state space that is searched (either progression or regression space), the search algorithm used (variants of best-first search), and the heuristic function extracted from the problem representation. This general planner implements a scheduler that tries different variants concurrently with different (time) resource bounds. We also describe how HSP2.0 can be used as an optimal (and near-optimal) planning algorithm and compare its performance with two other optimal planners, STAN and BLACKBOX.

HSP2.0 is a new version of the heuristic search planner (HSP), a domain-independent planner that was entered into the first planning competition (Long 2000; McDermott 2000). These planners, as well as the regression planner HSPR (Bonet and Geffner 1999) and the optimal planner HSPR* (Haslum and Geffner 2000), are all based on the same idea: Planning problems are mapped into search problems in a suitable space, which is solved using a heuristic function extracted automatically from the problem encoding. This formulation of planning as heuristic search appears in McDermott (1996) and Bonet, Loerincs, and Geffner (1997). Other heuristic search planners in the AIPS2000 Contest are GRT (Refanidis and Vlahavas 1999), FF (Hoffmann 2000), and ALTALT (Nguyen et al. 2000). Planners such as MIPS (Edelkamp and Helmart 2000) and STAN (Edelkamp and Helmart 2000) also make use of heuristic search ideas in certain contexts.

Pure heuristic search planners can be characterized along three main dimensions: (1) the state space that is searched (either the progression or regression space), (2) the search algorithm used (most often some version of best-first or hill-climbing search), and (3) the heuristic function extracted from the problem representation. The original version of HSP, for example, searches the progression space with a hill-climbing algorithm and a nonadmissible heuristic derived from a relaxation where delete lists are assumed empty, and atoms are assumed independent. FF searches the same progression space using a different hill-climbing algorithm and a different nonadmissible heuristic. Something similar is done by GRT. ALTALT, however, derives the heuristic from the plan graph constructed by a GRAPHPLAN-type procedure and uses this heuristic to drive a regression search from the goal. As a last example, HSPR* searches the regression space using the IDA* algorithm and an admissible heuristic function computed using a shortest-path algorithm.

Because no choice of the state space, search algorithm, and heuristic function appears best across different domains, we developed the HSP2.0 planner as a general platform for experimenting with different state spaces and different heuristics. The search algorithm in HSP2.0 is currently fixed and implements the WA* algorithm (Korf 1993; Pearl 1983), an A* algorithm in which the heuristic term $h(n)$ of the evaluation function $f(n) = g(n) + h(n)$ is multiplied by a parameter $W \geq 1$.

It is well known that the parameter $W$ achieves a trade-off between optimality and speed. For $W = 1$, WA* becomes A* and is optimal as long as the heuristic function is admissible (nonoverestimating). For higher $W$, for example, $W = 2$, WA* finds solutions faster, but these solutions are only guaranteed to be at most $W$ times away from optimal.

For a particular planning problem, the user of HSP2.0 can specify the state space to be searched, the heuristic, and the $W$ parameter.[1] For example, the invocation

```
hsp –d backward –h h2max –w 1.5
prob4.pddl domain.pddl
```

runs HSP over the regression space using the heuristic $h^2$ and the parameter $W = 1.5$ over the instance in file prob4.pddl with domain file domain.pddl. The syntax for the instance and domain files is given by the PDDL standard.[2] HSP2.0 deals with a fragment of PDDL that includes STRIPS, negation, types, and conditional effects.

In HSP2.0, the search can be done either forward or backward, and the heuristics can be the nonadmissible heuristic $h_{add}$ used in the original version of HSP, the admissible heuristic $h_{max}$, and the more informed admissible heuristic $h^2$ formulated in Haslum and Geffner (2000). By default, HSP2.0 searches in the forward direction with $h = h_{add}$ and $W = 2$. The user can also choose a schedule of options as done in BLACKBOX. For example, it can tell HSP2.0 to use some direction and heuristic for a fixed amount of time and then switch to a different direction-heuristic combination if no solution is found. In addition, the different options can be run concurrently as threads. In the Fifth International Conference on Artificial Intelligence Planning and Scheduling (AIPS'00), HSP2.0 ran three options concurrently for three minutes: (1) forward/$h_{add}$, backward/$h_{add}$, and backward/$h^2$, all with $W = 2$. If no solution was reported by then, HSP2.0 continued with the forward/$h_{add}$ option only. The choice for the three-minute threshold was empirical, following the observation that the regression search most often solved problems quickly or didn't solve them at all. The curve for the forward/$h_{add}$ combination is smoother: More time often means more problems solved. Some options clearly dominated others in particular domains: For example, in the logistics domain, most problems were solved by the backward/$h_{add}$ option, but in the *FreeCell* domain most problems were solved by forward/$h_{add}$.

In the rest of this article, we make more precise the options supported by the HSP2.0 planner, illustrate the optimality-speed trade-off involved in the use of the $W$ parameter, and briefly discuss some results.

## State Spaces

A STRIPS problem is a tuple $P = \langle A, O, I, G\rangle$, where $A$ is a set of atoms, $O$ is a set of ground STRIPS operators, and $I \subseteq A$ and $G \subseteq A$ encode the initial and goal situations. The state space determined by $P$ is a tuple $S = \langle S, s_0, S_G, A(\cdot), f, c\rangle$, where

S1. The states $s \in S$ are collections of atoms from $A$.

S2. The initial state $s_0$ is $I$.

S3. The goal states $s \in S_G$ are such that $G \subseteq s$.

S4. The actions $a \in A(s)$ are the operators $op \in O$ such that $Prec(op) \subseteq s$.

S5. The transition function $f$ maps states $s$ into states $s' = s - Del(a) + Add(a)$ for $a \in A(s)$.

S6. The action costs $c(a)$ are all equal to 1.

We refer to this state space as the *progression space*, in contrast to the regression space described later. When HSP is told to search in the forward direction, it searches the progression space starting from the initial state $s_0$.

The *regression state space* associated with the same planning problem $P$ can be defined by the tuple $\Re = (S, s_0, S_G, A(\cdot), f, c)$, where

R1. The states $s \in S$ are sets of atoms from $A$.

R2. The initial state $s_0$ is the goal $G$.

R3. The goal states $s \in S_G$ are such that $s \subseteq I$.

R4. The set of actions $A(s)$ applicable in $s$ are the operators $op$ in $O$ that are relevant and consistent; namely, $Add(op) \cap s \neq \emptyset$, and $Del(op) \cap s = \emptyset$.

R5. The state $s_0 = f(a,s)$ that follows from the application of $a = op$ in $s$, for $a \in A(s)$, is such that $s' = s - Add(o\text{p}) + Prec(op)$.

R6. The action costs $c(a)$ are all equal to 1.

HSP2.0 searches this regression space when told to search backward from the goal. The regression search has the advantage that it makes it possible to compute the bulk of the heuristic function only once (Bonet and Geffner 1999). In the progression search, the heuristic values are computed from scratch in every new state. This recomputation is expensive, yet in certain cases, it pays off by providing useful new information. Likewise, the regression search has the problem that it often generates spurious states: states that are not reachable from the initial problem situation, such as those where one block is on top of two different blocks. Some such states are detected by a procedure that identifies pairs of mutually exclusive propositions adapted from GRAPHPLAN (Blum and Furst 1997). Overall, the forward search is slower than the regression search in certain domains, but in general, it appears to be more robust (Bonet and Geffner 2000).

## Heuristics

The heuristics $h_{add}$ and $h_{max}$ in HSP are derived as approximations of the optimal cost function of a "relaxed" planning problem in which delete

lists are ignored. More precisely, the heuristics are obtained by combining the estimated costs $g(p; s)$ of achieving each of the goal atoms $p$ from a state $s$. These estimates are obtained by solving the functional equation

$$g(p;s) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } p \in s \\ \min_{a \in O(p)} \left[1 + g(\text{Prec}(a); s)\right] & \text{otherwise} \end{cases} \quad (1)$$

for all atoms $p$ by means of a Bellman-Ford type of algorithm. In this algorithm, the measures $g(p; s)$ are updated as

$$g(p;s) := \min_{a \in O(p)} \left[g(p;s), 1 + g(\text{Prec}(a); s)\right] \quad (2)$$

starting with $g(p; s) = 0$ if $p \in s$ and $g(p; s) = \infty$ otherwise, until they do not change. In these expressions, $O(p)$ stands for the set of operators that "add" $p$, and $g(Prec(a); s)$ stands for the estimated cost of achieving the set of atoms $Prec(a)$ from $s$.

For the additive heuristic $h_{add}$, the cost $g(C; s)$ of sets of atoms $C$ is defined as the sum of the costs $g(r; s)$ of the individual atoms $r$ in $C$. We denote such additive costs as $g_{add}(C; s)$:

$$g_{add}(C;s) \stackrel{\text{def}}{=} \sum_{r \in C} g(r;s) \quad \text{(additive costs)} \quad (3)$$

The heuristic $h_{add}(s)$ is then defined as

$$h_{add}(s) \stackrel{\text{def}}{=} g_{add}(G;s) \quad (4)$$

The definition of the cost of sets of atoms in equation 3 assumes that subgoals are independent, which is not true in general, and as a result, the heuristic might overestimate costs and is not admissible. An admissible heuristic can be obtained by defining the costs $g(C; s)$ of sets of atoms as

$$g_{\max}(C;s) \stackrel{\text{def}}{=} \max_{r \in C} g(r;s) \quad \text{(max costs)} \quad (5)$$

The resulting max heuristic $h_{max}(s) = g_{max}(G; s)$ is admissible, but it is not very informative. Both heuristics $h_{add}$ and $h_{max}$ are supported in HSP2.0.

The last heuristic supported in HSP2.0 is an admissible heuristic $h^2$ that dominates $h_{max}$. Both $h^2$ and $h_{max}$ can be understood as instances of a general family of polynomial and admissible heuristics $h^m$ for $m = 1, 2, \ldots$ (Haslum and Geffner 2000). The higher the value of $m$, the more accurate the heuristic but the more expensive its calculation. The idea underlying the heuristics $h^m$ is to approximate the cost $g(C; s)$ for achieving a set of atoms $C$ from a state $s$ by the cost $g^m(C; s)$ of the most costly subset of size $m$ in $C$. Thus, for $m = 1$, the cost $g(C; s)$ is approximated by the cost of the most costly atom in $C$, for $m = 2$, by the cost of the most costly atom pair in $C$, and so on. The costs $g^m(C; s)$ are characterized by the equation in figure 1, where $B \in R(C)$ if $B$ is the result of regressing the set of atoms $C$ through some action $a$. The heuristic function $h^m(s)$ is then defined as $g^m(G; s)$, where $G$ is the goal. HSP2.0 accommodates the heuristics $h^m$ for $m = 1$ and $m = 2$ only. Both are obtained by solving the equation in figure 1 with a Bellman-Ford type of shortest-path algorithm.

$$g^m(C;s) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } C \subseteq s, \text{ else} \\ \min_{B \in R(C)} \left[1 + g^m(B;s)\right] & \text{if } |C| \leq m \\ \max_{D \subset C, |D| = m} g^m(D;s) & \text{otherwise} \end{cases}$$

*Figure 1. The Costs $g^m(C; s)$.*

## Results

As mentioned earlier, in the competition, HSP2.0 was run with three concurrent options for three minutes: (1) forward/$h_{add}$, (2) backward/$h_{add}$, and (3) backward/$h^2$. If no solution was found by then, only the option forward/$h_{add}$ was allowed to continue. HSP2.0 solved all problems in most of the domains, except for the scheduling domain, where it solved the smallest instances only, and blocks world, where it failed to solve some of the large instances. In many cases, such as in blocks world, the plans generated were long and far from optimal. Overall, the planner FF, based on similar ideas, did better and, in certain cases, significantly better primarily because of three factors: (1) the search direction (forward search appears to be more robust than backward search), (2) a heuristic more accurate than $h_{add}$ (the only heuristic used by HSP2.0 in the forward direction), and (3) a good rule for quickly discarding nodes without computing their heuristic values. These features, and others, discussed at more length in the article by Jörg Hoffmann, appear to make FF a more powerful planner than HSP. A potential advantage of HSP is that it uses a more systematic search algorithm and that it can be used to generate optimal or arbitrarily close to optimal plans. Optimal plans are obtained for the settings $h = h^2$, dir = backward, and $W = 1$. At the same time, if a value of $W > 1$ is used, the resulting plans are known to be at most $W$ times away from optimal, and they are found much faster. As an optimal planner, HSP2.0 appears to be competitive with optimal (parallel) GRAPHPLAN and SAT planners such as STAN and BLACKBOX (see table 1). At the same time, plans can be obtained much faster with a small degradation in plan quality by using a value of $W$ slightly higher than 1 (see table 2). This trade-off is not so easily available in either the GRAPHPLAN or SAT planners.

| | HSP | | STAN | | BLACKBOX | |
|---|---|---|---|---|---|---|
| *Problem* | *Length* | *Time* | *Length* | *Time* | *Length* | *Time* |
| 7-0 | 20 | 0.18 | 20 | 0.05 | 20 | 0.28 |
| 7-1 | 22 | 0.19 | 22 | 0.13 | 22 | 0.78 |
| 7-2 | 20 | 0.17 | 20 | 0.08 | 20 | 0.43 |
| 8-0 | 18 | 0.18 | 19 | 0.13 | 18 | 0.79 |
| 8-1 | 20 | 0.20 | 20 | 0.15 | 20 | 1.16 |
| 8-2 | 16 | 0.18 | 16 | 0.07 | 16 | 0.46 |
| 9-0 | 30 | 0.44 | 30 | 0.29 | 30 | 3.20 |
| 9-1 | 28 | 0.21 | 28 | 0.17 | 28 | 1.21 |
| 9-2 | 26 | 0.25 | 26 | 0.14 | 26 | 0.89 |
| 10-0 | 34 | 1.37 | 34 | 0.95 | 34 | 7.78 |
| 10-1 | 32 | 361.19 | 32 | 24.24 | 32 | 220.03 |
| 10-2 | 34 | 1.61 | 34 | 1.44 | 34 | 19.35 |
| 11-0 | 32 | 333.72 | 32 | 346.94 | 32 | 409.47 |
| 11-1 | — | — | — | — | 30 | 504.61 |
| 11-2 | 34 | 38.65 | 34 | 89.96 | 34 | 114.42 |
| 12-0 | 34 | 3.99 | 34 | 15.98 | 34 | 50.30 |
| 12-1 | 34 | 1.00 | 34 | 2.39 | 34 | 24.52 |

*Table 1. Optimal Planning.*

Results over blocks-world instances from competition for some optimal planners. hsp2.0 results obtained with options $d$ = backward, $h = h^2$, and $W = 1$. Time reported in seconds.

| | HSP, W = 1 | | HSP, W = 1.25 | | HSP, W = 1.75 | |
|---|---|---|---|---|---|---|
| *Problem* | *Length* | *Time* | *Length* | *Time* | *Length* | time |
| 7-0 | 20 | 0.18 | 20 | 0.17 | 22 | 0.17 |
| 7-1 | 22 | 0.19 | 22 | 0.18 | 22 | 0.21 |
| 7-2 | 20 | 0.17 | 20 | 0.17 | 20 | 0.18 |
| 8-0 | 18 | 0.18 | 18 | 0.18 | 18 | 0.17 |
| 8-1 | 20 | 0.20 | 20 | 0.21 | 20 | 0.18 |
| 8-2 | 16 | 0.18 | 16 | 0.22 | 16 | 0.18 |
| 9-0 | 30 | 0.44 | 32 | 0.36 | 32 | 0.35 |
| 9-1 | 28 | 0.21 | 28 | 0.25 | 30 | 0.21 |
| 9-2 | 26 | 0.25 | 26 | 0.24 | 26 | 0.19 |
| 10-0 | 34 | 1.37 | 36 | 0.57 | 40 | 0.53 |
| 10-1 | 32 | 361.19 | 32 | 18.48 | 34 | 0.37 |
| 10-2 | 34 | 1.61 | 34 | 0.59 | 38 | 0.23 |
| 11-0 | 32 | 333.72 | 32 | 2.60 | 34 | 0.28 |
| 11-1 | — | — | — | — | — | — |
| 11-2 | 34 | 38.65 | 34 | 0.62 | 34 | 0.65 |
| 12-0 | 34 | 3.99 | 34 | 0.53 | 34 | 0.31 |
| 12-1 | 34 | 1.00 | 34 | 0.31 | 36 | 0.29 |

*Table 2. Optimality-Speed Trade-Off.*

Results for HSP2.0 over same instances but with different $W$ values.

## Note

1. HSP2.0 is available at www.ldc.usb.ve/~hector and www.cs.ucla.edu/~bonet.

2. D. McDermott, 1998, PDDL, the planning domain definition language. Available at www.cs.yale.edu/~dvm.

## References


Blum, A. L., and Furst, M. L. 1997. Fast Planning through Planning Graph Analysis. *Artificial Intelligence* 90(1–2): 281–300.

Bonet, B., and Geffner, H. 2001. Planning as Heuristic Search (Special Issue on Heuristic Search). *Artificial Intelligence* 129(1–2): 5–33.

Bonet, B., and Geffner, H. 1999. Planning as Heuristic Search: New Results. In *Recent Advances in AI Planning: Proceedings of the Fifth European Conference on Planning,* 359–371. Lecture Notes in AI 1809. New York: Springer.

Bonet, B.; Loerincs, G; and Geffner, H. 1997. A Robust and Fast Action Selection Mechanism for Planning. In Proceedings of the Fourteenth National Conference on Artificial Intelligence, 714–719. Menlo Park, Calif.: American Association for Artificial Intelligence.

Edelkamp, S., and Helmert, M. 2000. On the Implementation of MIPS. Paper presented at the Workshop on Model-Theoretic Approaches to Planning, Fifth International Conference on Artificial Intelligence Planning and Scheduling (AIPS'00), 14–17 April, Breckenridge, Colorado.

Haslum, P., and Geffner, H. 2000. Admissible Heuristics for Optimal Planning. In *Proceedings of the Fifth International Conference on AI Planning Systems,* 70–82. Menlo Park, Calif.: AAAI Press.

Hoffmann, J. 2000. A Heuristic for Domain-Independent Planning and Its Use in an Enforced Hill-Climbing Algorithm. In *Proceedings of the Twelfth International Symposium on Methodologies for Intelligent Systems (ISMIS-00),* 216–227. New York: Springer.

Korf, R. 1993. Linear-Space Best-First Search. *Artificial Intelligence* 62(1): 41–48.

Long, D. 2000. The AIPS-98 Planning Competition. *AI Magazine* 21(2): 13–34.

Long, D., and Fox, M. 1999. The Efficient Implementation of the Plan-Graph in STAN. *Journal of Artificial Intelligence Research* 10:85-115.

McDermott, D. 2000. The 1998 AI Planning Systems Competition. *AI Magazine* 21(2): 35–56.

McDermott, D. 1996. A Heuristic Estimator for Means-Ends Analysis in Planning. In *Proceedings of the Third International Conference on AI Planning Systems,* 142–149, Menlo Park, Calif.: AAAI Press.

Pearl, J. 1983. *Heuristics*. San Francisco, Calif.: Morgan Kaufmann.

Nguyen, Z.; Nigenda, R. S.; and Kambhampati, S. 2000. ALTALT: Combining the Advantages of GRAPHPLAN and Heuristic State Search. Technical report, College of Engineering and Applied Sciences, Arizona State University.

Refanidis, I., and Vlahavas, I. 1999. GRT: A Domain-Independent Heuristic for STRIPS Worlds Based on Breedy Regression Tables. In *Recent Advances in AI Planning: Proceedings of the Fifth European Conference on Planning,* 346–358. Lecture Notes in AI 1809. New York: Springer.



**Blai Bonet** got his bachelor's degree and M.S. in computer science from Universidad Simon Bolivar in Caracas, Venezuela, under the supervision of Hector Geffner. He has published papers in the areas of classical planning, planning with uncertainty and incomplete information, and causal and probabilistic reasoning. He is currently a Ph.D. student at the University of California at Los Angeles. His e-mail address is bonet@ldc.usb.ve.

**Hector Geffner** got his Ph.D. at the University of California at Los Angeles with a dissertation that was co-winner of the 1990 Association for Computing Machinery Dissertation Award. He worked as staff research member at the IBM T.J. Watson Research Center in New York for two years before returning to the Universidad Simon Bolivar, in Caracas, Venezuela, where he currently teaches. Geffner is author of the book *Default Reasoning: Causal and Conditional Theories,* which was published by MIT Press in 1992. He is interested in models of reasoning, action, and learning and in the development of high-level tools for modeling and problem solving. His e-mail address is hector@ldc.usb.ve.