Generalized Best-First Search Strategies and the Optimality of A*

RINA DECHTER AND JUDEA PEARL

University of California, Los Angeles, California

Abstract. This paper reports several properties of heuristic best-first search strategies whose scoring functions f depend on all the information available from each candidate path, not merely on the current cost g and the estimated completion cost h. It is shown that several known properties of A* retain their form (with the minmax of f playing the role of the optimal cost), which helps establish general tests of admissibility and general conditions for node expansion for these strategies. On the basis of this framework the computational optimality of A*, in the sense of never expanding a node that can be skipped by some other algorithm having access to the same heuristic information that A* uses, is examined. A hierarchy of four optimality types is defined and three classes of algorithms and four domains of problem instances are considered. Computational performances relative to these algorithms and domains are appraised. For each class–domain combination, we then identify the strongest type of optimality that exists and the algorithm for achieving it. The main results of this paper relate to the class of algorithms that, like A*, return optimal solutions (i.e., admissible) when all cost estimates are optimistic (i.e., $h \leq h^*$). On this class, A* is shown to be not optimal and it is also shown that no optimal algorithm exists, but if the performance tests are confirmed to cases in which the estimates are also consistent, then A* is indeed optimal. Additionally, A* is also shown to be optimal over a subset of the latter class containing all *best-first* algorithms that are guided by path-dependent evaluation functions.

Categories and Subject Descriptors: I.2.8 **[Artificial Intelligence]**: Problem Solving, Control Methods and Search—*graph and tree search strategies*

General Terms: Algorithms, Measurement, Performance, Theory

Additional Key Words and Phrases: Best-first strategies, branch and bound, heuristic search, shortest path algorithms

1. *Introduction*

Of all search strategies used in problem solving, one of the most popular methods of exploiting heuristic information to cut down search time is the *informed best-first* strategy. The general philosophy of this strategy is to use the heuristic information to assess the "merit" latent in every candidate search avenue exposed during the search and then continue the exploration along the direction of highest merit. Formal descriptions of this strategy are usually given in the context of *path-searching problems* [16], a formulation that represents many combinatorial problems, such as routing, scheduling, speech recognition, and scene analysis.

Given a weighted directional graph G with a distinguished start node s and a set of goal nodes Γ, the *optimal path problem* is to find a least-cost path from s to any

This work was supported in part by the National Science Foundation under Grant MCS 81-14209.

Authors' address: Cognitive Systems Laboratory, Computer Science Department, University of California, Los Angeles, CA 90024.

member of Γ where the cost of the path may, in general, be an arbitrary function of the weights assigned to the nodes and branches along that path. A *general best-first* (GBF) strategy will pursue this problem by constructing a tree T of selected paths of G using the elementary operation of *node expansion*, that is, generating all successors of a given node. Starting with s, GBF will select for expansion that leaf node of T that features the highest "merit," and will maintain in T all previously encountered paths that still appear as viable candidates for sprouting an optimal solution path. The search terminates when no such candidate is available for further expansion, in which case the best solution path found so far is issued as a solution; if none has been found, a failure is proclaimed.

In practice, several shortcuts have been devised to simplify the computation of GBF. First, if the evaluation function used for node selection always provides optimistic estimates of the final costs of the candidate paths evaluated, then we can terminate the search as soon as the first goal node is selected for expansion without compromising the optimality of the solution issued. This guarantee is called *admissibility* and is, in fact, the basis of the branch-and-bound method [10]. Second, we are often able to purge from T large sets of paths that are recognized at an early stage to be *dominated* by other paths in T [8]. This becomes particularly easy if the evaluation function f is *order preserving*, that is, if, for any two paths P_1 and P_2, leading from s to n, and for any common extension P_3 of those paths, the following holds:

$$f(P_1) \geq f(P_2) \Rightarrow f(P_1\ P_3) \geq f(P_2\ P_3).$$

Order preservation is a judgmental version of the so-called *principle of optimality* in Dynamic Programming [3], and it simply states that, if P_1 is judged to be more meritorious than P_2, both going from s to n, then no common extension of P_1 and P_2 may later reverse this judgment. Under such conditions, there is no need to keep in T multiple copies of nodes of G; each time the expansion process generates a node n that already resides in T, we maintain only the lower f path to it, discarding the link from the more expensive father of n.[1]

These two simplifications are implemented by the following best-first algorithm, a specialization of GBF, which we call BF* [16]:

*Algorithm BF**

1. Put the start node s on a list called OPEN of unexpanded nodes.
2. If OPEN is empty, exit with failure; no solution exists.
3. Remove from OPEN a node n at which f is minimum (break ties arbitrarily, but in favor of a goal node), and place it on a list called CLOSED to be used for expanded nodes.
4. If n is a goal node, exit successfully with the solution obtained by tracing back the path along the pointers from n to s (pointers are assigned in Steps 5 and 6).
5. Expand node n, generating all its successors with pointers back to n.
6. For every successor n' of n:
   a. Calculate $f(n')$.
   b. If n' was neither in OPEN nor in CLOSED, then add it to OPEN. Assign the newly computed $f(n')$ to node n'.
   c. If n' already resided in OPEN or CLOSED, compare the newly computed $f(n')$ with that previously assigned to n'. If the new value is lower, substitute it for the old (n' now points back to n instead of to its predecessor). If the matching node n' resided in CLOSED, move it back to OPEN.
7. Go to Step 2.

[1] If f is not order preserving, one may sometimes use an auxiliary function to perform pruning by dominance while f is used strictly for ranking the candidates for expansion. Such a use of two separate functions is excluded from our formulation of best first search.

By far, the most studied version of BF* is the algorithm A* [6], which was developed for *additive cost measures*, that is, where the cost of a path is defined as the sum of the costs of its arcs. To match this cost measure, A* employs an additive evaluation function $f(n) = g(n) + h(n)$, where $g(n)$ is the cost of the currently evaluated path from s to n and h is a heuristic estimate of the cost of the path remaining between n and some goal node. Since $g(n)$ is order preserving and $h(n)$ depends only on the description of the node n, $f(n)$ too is order preserving, and one is justified in discarding all but one parent for each node, as in step 6c of BF*. If, in addition, $h(n)$ is a lower bound to the cost of any continuation path from n to Γ, then $f(n)$ is an optimistic estimate of all possible solutions containing the currently evaluated path, and terminating A* upon the selection of the first goal node (step 4) does not compromise its admissibility.[2] Several other properties of A* can be established if admissibility holds, such as the conditions for node expansion, node reopening, and the fact that the number of nodes expanded decreases with increasing h [14, results 5, 6, and 7]. These properties are essential for any quantitative analysis of the performance of A* [16].

It has been found, however, that maintaining the admissibility of A* is too restrictive; it limits the selection of heuristics to those that only underestimate costs, and it forces A* to spend a disproportionately long time discriminating among roughly equal candidates. As a result, several attempts have been made to execute A* with nonadmissible estimates while simultaneously limiting the degree of suboptimality [5, 17, 18, 20] and minimizing the computational difficulties that overestimates may induce [1]. However, most of our knowledge regarding the behavior of A* is still limited to additive cost measures and additive evaluation functions.

In this paper, our aim is to examine how the behavior of A* will change if we remove both restrictions. The cost minimization criterion is generalized to include nonadditive cost measures of solution paths, such as multiplicative costs, the *max*-cost (i.e., the highest branch cost along the path), the *mode* (i.e., the most frequent branch cost along the path), the *range* (i.e., the difference between the highest and the lowest branch costs along the path), the cost of the *last* branch, and the *average* cost. Additionally, even in cases in which the minimization objective is an additive cost measure, we now permit $f(n)$ to take on a more general form and to employ more elaborate evaluations of the promise featured by a given path from s to n, utilizing all the information gathered along that path. For example, one may wish to consult the evaluation function $f(n) = \max_n\{g(n') + h(n')\}$ where n' ranges along the path from s to n. Alternatively, the class of evaluation functions may now include nonlinear combinations of g and h in $f = f(g, h)$ and, as a very special example, the additive combination $f = g + h$ with h an arbitrary heuristic function, not necessarily optimistic.

We start by characterizing the performance of the algorithm BF* without assuming any relationship between the cost measure C, defined on complete solution paths and the evaluation function f, defined on partial paths. Later on, assuming a monotonic relationship between f and C on complete solution paths, we establish a relationship between the cost of the solution found and that of the

[2] Our definition of A* is identical to that of Nilsson [13] and is at variance with Nilsson's later version [14]. The latter regards the requirement $h \leq h^*$ as part of the definition of A*; otherwise the algorithm is called A. We found it more convenient to follow the tradition of identifying an algorithm by how it processes input information rather than by the type of information it may encounter. Accordingly, we assign the symbol A* to any BF* algorithm that uses the additive combination $f = g + h$, placing no restriction on h, in line with the more recent literature [1, 2, 16].

optimal solution (Section 2). In Section 3, we establish conditions for node expansion that could be used for analyzing the performance of BF* algorithms. Finally, in Section 4, we consider the performance of A* under the additive cost measure and examine under what conditions A* (employing $f = g + h$) is computationally optimal over other search algorithms that are provided with the same heuristic information h and are guaranteed to find solutions comparable to those found by A*.

For simplicity, we shall present our analysis relative to the BF* algorithm with the assumption that f is order preserving. However, all of our results hold for evaluation functions that are not order preserving if, instead of BF*, GBF is executed; namely, all paths leading to a given node are maintained and evaluated separately.

Notation

G	Directed locally finite graph, $G = (V, E)$.
C^*	The cost of the cheapest solution path.
$C(\cdot)$	The cost function defined over all solution paths.
Γ	A set of goal nodes, $\Gamma \subseteq V$.
$P_{n_i-n_j}$	A path in G between node n_i and n_j.
P^s	A *solution path*, that is, a path in G from s to some goal node $\gamma \in \Gamma$.
$c(n, n')$	The cost of an arc between n and n', $c(n, n') \geq \delta > 0$, where δ is a constant.
$f(\cdot)$	Evaluation function defined over partial paths; that is, to each node n along a given path $P = s_1, n_1, n_2, \ldots, n$ we assign the value $f_P(n)$, which is a shorthand notation for $f(s, n_1, n_2, \ldots, n)$.
$g(n)$	The sum of the branch costs along the current path of pointers from n to s.
$g^*(n)$	The cost of the cheapest path going from s to n.
$g_P(n)$	The sum of the branch costs along path P from s to n.
$h(n)$	A cost estimate of the cheapest path remaining between n and Γ.
$h^*(n)$	The cost of the cheapest path going from n to Γ.
$k(n, n')$	The cost of the cheapest path between n and n'.
M	The minimal value of M_j over all j.
M_j	The highest value of $f_P(n)$ along the jth solution path.
n^*	A node for which $f(\cdot)$ attains the value M.
s	Start node.
T	A subtree of G containing all the arcs to which pointers are currently assigned.

For the sake of comparison, we now quote some basic properties of A* [14, 16], which we later generalize.

Result 1. If $h \leq h^*$, then at any time before A* terminates there exists on OPEN a node n' that is on an optimal path from s to a goal node, with $f(n') \leq C^*$.

Result 2. If there is a path from s to a goal node, A* terminates for every $h \geq 0$ (G can be infinite).

Result 3. If $h \leq h^*$, then algorithm A* is admissible (i.e., it finds an optimal path).

Result 4. If $h \leq h^*$, then any time a node n is selected for expansion by A* it must satisfy $f_P(n) \leq C^*$, where P is the pointer path assigned to n at the time of expansion.

Result 5. If $h \leq h^*$, then every node in OPEN for which $f(n) < C^*$ will eventually be expanded by A*.

2. *Algorithmic Properties of Best-first (BF*) Search*

In locally finite graphs, the set of solution paths is countable, so they can be enumerated:

$$P_1^S, P_2^S, \ldots, P_j^S, \ldots,$$

and, correspondingly, we can use the notation $f_j(n)$ to represent $f_{P_j^S}(n)$. Let M_j be the maximum of f on the solution path P_j^S, that is,

$$M_j = \max_{n \in P_j^S} \{f_j(n)\};$$

and let M be the minimum of all the M_j's:

$$M = \min_j \{M_j\}.$$

The minmax value M can be interpreted as the level of the "saddle point" in the network of paths leading from s to Γ. We henceforth assume that both the max and the min functions are well defined.

LEMMA 1. *If BF* uses an order-preserving f, and P_1 and P_2 are any two paths from s to n such that the pointers from n to s currently lie along P_1 and all the branches of P_2 have been generated in the past, then*

$$f_{P_1}(n) \leq f_{P_2}(n).$$

In particular, if P_t is known to be the path assigned to n at termination, then

$$f_{P_t}(n) \leq f_P(n) \qquad \forall P \in G_e,$$

where G_e is the subgraph explicated during the search, namely, the union of all past traces of T.

PROOF. The lemma follows directly from the order-preserving properties of f and the fact that pointers are redirected toward the path yielding a lower f value. The former guarantees that, if at least one node on P_2 was ever assigned a pointer directed along an alternative path superior to the direction of P_2, then the entire P_2 path to n will remain inferior to every alternative path to n subsequently exposed by BF*. □

2.1 TERMINATION AND COMPLETENESS

LEMMA 2. *At any time before BF* terminates, there exists on OPEN a node n' that is on some solution path and for which $f(n') \leq M$.*

PROOF. Let $M = M_j$, that is, the minmax is obtained on solution path P_j^S. Then, at some time before termination, let n' be the shallowest OPEN node on P_j^S having pointers directed along P_i^S (possibly $i = j$). From the definition of M_j,

$$M_j = \max_{n \in P_j^S} \{f_j(n)\};$$

therefore,

$$f_j(n') \leq M_j = M.$$

Moreover, since all ancestors of n' on P_j^S are in CLOSED and BF* has decided to assign its pointers along P_i^S, Lemma 1 states that

$$f_i(n') \leq f_j(n').$$

This implies

$$f_i(n') \leq M,$$

which proves Lemma 2. □

LEMMA 3. *Let P_j^S be the solution path with which BF* terminates; then*

(a) *any time before termination there is an OPEN node n on P_j^S for which $f(n) = f_j(n)$;*
(b) *M is obtained on P_j^S, that is, $M = M_j$.*

PROOF

(a) Let n be the shallowest OPEN node on P_j^S at some arbitrary time t, before termination. Since all n's ancestors on P_j^S are closed at time t, n must be assigned an f at least as cheap as $f_j(n)$. Thus, $f_j(n) \geq f(n)$ with strict inequality holding only if, at time t, n is found directed along a path different than P_j^S. However, since BF* eventually terminates with pointers along P_j^S, it must be that BF* has never encountered another path to n with cost lower than $f_j(n)$. Thus, $f(n) = f_j(n)$.

(b) Suppose BF* terminates on P_j^S, but $M_j > M$, and let $n^* \in P_j^S$ be a node where $f_j(\cdot)$ attains its highest value, that is, $f_j(n^*) = M_j$. At the time that n^* is last chosen for expansion, its pointer must already be directed along P_j^S and, therefore, n^* is assigned the value $f(n^*) = f_j(n^*) > M$. At that very time there exists an OPEN node n' having $f(n') \leq M$ (Lemma 2), and so

$$f(n') < f(n^*).$$

Accordingly, n' should be expanded before n^*, which contradicts our supposition. □

THEOREM 1. *If there is a solution path and f is such that $f_P(n)$ is unbounded along any infinite path P, then BF* terminates with a solution, that is, BF* is complete.*

PROOF. In any locally finite graph, there is only a finite number of paths with finite length. If BF* does not terminate, then there is at least one infinite path along which every finite-depth node will eventually be expanded. That means that f must increase beyond bounds and, after a certain time t, no OPEN nodes on any given solution path will ever be expanded. However, from Lemma 2, $f(n') \leq M$ for some OPEN node n' along a solution path, which contradicts the assumption that n' will never be chosen for expansion. □

The condition of Theorem 1 is clearly satisfied for additive cost measures owing to the requirement that each branch cost be no smaller than some constant δ. It is also satisfied for many quasi-additive cost measures (e.g., $[\sum_i c_i^r]^{1/r}$), but may not hold for "saturated" cost measures such as the maximum cost. In the latter cases, termination cannot be guaranteed on infinite graphs and must be controlled by such special means as using iteratively increasing depth bounds.

We shall soon see that the importance of M lies not only in guaranteeing the termination of BF* on infinite graphs but mainly in identifying and appraising the solution path eventually found by BF*.

2.2 QUALITY OF THE SOLUTION FOUND, AND THE ADMISSIBILITY OF BF*. So far we have not specified any relation between the cost function C, defined on solution paths, and the evaluation function f, defined on partial paths or solution candidates. Since the role of f is to guide the search toward the lowest cost solution path, we now impose the restriction that f be monotonic with C when evaluated on complete solution paths; that is, if P and Q are two solution paths with $C(P) > C(Q)$, then $f(P) > f(Q)$. Since $C(P)$ and $C(Q)$ can take on arbitrarily close values over our space of problem instances, monotonicity can be represented by

$$f(s, n_1, n_2, \ldots, \gamma) = \psi[C(s, n_1, n_2, \ldots, \gamma)] \qquad \gamma \in \Gamma, \tag{1}$$

where ψ is some increasing function of its argument defined over the positive reals. No restriction, however, is imposed on the relation between C and f on nongoal nodes; that is, we may allow evaluation functions that treat goal nodes preferentially, for example,

$$f(s, n_1, n_2, \ldots, n) = \begin{cases} \psi[C(s, n_1, n_2, \ldots, n)] & \text{if } n \in \Gamma, \\ F(s, n_1, n_2, \ldots, n) & \text{if } n \notin \Gamma, \end{cases}$$

where $F(\cdot)$ is an arbitrary function of the path $P = s, n_1, n_2, \ldots, n$. The additive evaluation function $f = g + h$ used by A* is, in fact, an example of such goal-preferring types of functions; the condition $h(\gamma) = 0$ guarantees the identity $f = C$ on solution paths, whereas on other paths f is not governed by C, since, in general, h could take on arbitrary values. Other examples of goal-preferring f evolve in branch-and-bound methods of solving integer-programming problems where the quality of a solution path is determined solely by the final goal node reached by that path. Here $f(n)$ may be entirely arbitrary for nongoal nodes, but if n is a goal node, we have $f(n) = C(n)$.

We now give some properties of the final solution path using the relationship stated above.

THEOREM 2. *BF* is $\psi^{-1}(M)$-admissible, that is, the cost of the solution path found by BF* is at most $\psi^{-1}(M)$.*

PROOF. Let BF* terminate with solution path $P_j^S = s, \ldots, t$ where $t \in \Gamma$. From Lemma 2, we learn that BF* cannot select for expansion any node n having $f(n) > M$. This includes the node $t \in \Gamma$ and, hence, $f_j(t) \leq M$. But (1) implies that $f_j(t) = \psi[C(P_j^S)]$ and so, since ψ and ψ^{-1} are monotonic,

$$C(P_j^S) \leq \psi^{-1}(M),$$

which proves the theorem. □

A similar result was established by Bagchi and Mahanti [1], although they used $\psi(C) = C$ and restricted f to the form $f = g + h$.

Theorem 2 can be useful in appraising the degree of suboptimality exhibited by nonadmissible algorithms. For example, Pohl [20] suggests a dynamic weighting scheme for the evaluation function f. In his approach the evaluation function f is given by

$$f(n) = g(n) + h(n) + \epsilon\left[1 - \frac{d(n)}{N}\right] h(n), \tag{2}$$

where $d(n)$ is the depth of node n and N is the depth of the shallowest optimal goal node. Using Theorem 2, we can easily show that, if $h(n) \leq h^*(n)$, then BF* will always find a solution within a $(1 + \epsilon)$ cost factor of the optimal, a property first

shown by Pohl [20]. In this case, the requirement $h(\gamma) = 0$ for $\gamma \in \Gamma$ and eq. (1) together dictate $\psi(C) = C$, and we can bound M by considering $\max_n f_{P^*}(n)$ along any optimal path P^*. Thus,

$$\begin{aligned} M &\leq \max_{n \in P^*} f_{P^*}(n) \\ &\leq \max_{n \in P^*} \left[g^*(n) + h^*(n) + \epsilon h^*(n) \left[1 - \frac{d(n)}{N} \right] \right] \\ &= C^* + \epsilon h^*(s) \\ &= C^*(1 + \epsilon). \end{aligned}$$

On the other hand, according to Theorem 2, the search terminates with cost $C_t \leq M$; hence

$$C_t \leq C^*(1 + \epsilon).$$

This example demonstrates that any evaluation function of the form

$$f(n) = g(n) + h(n)[1 + \epsilon \rho_P(n)]$$

will also return a cost at most $(1 + \epsilon)C^*$, where $\rho_P(n)$ is an arbitrary path-dependent function of node n, satisfying $\rho_P(n) \leq 1$ for all n along some optimal path P^*. For example, the functions

$$\rho_P(n) = \left[\frac{h(n)}{g(n) + h(n)} \right]^K \quad \text{or} \quad \rho_P(n) = [1 + d(n)]^{-K}, \quad K > 0,$$

will qualify as replacements for $[1 - d(n)/N]$.

The main utility of Theorem 2, however, lies in studying search on graphs with *random* costs where we can use estimates of M to establish probabilistic bounds on the degree of suboptimality, $C_t - C^*$. An example of such an option arises in the problem of finding the cheapest root-to-leaf path in a uniform binary tree of height N, where each branch independently may have a cost of 1 or 0 with probability p and $1 - p$, respectively.

It can be shown [9] that for $p > \frac{1}{2}$ and large N the optimal cost is very likely to be near $\alpha^* N$ where α^* is a constant determined by p. Consequently, a natural evaluation function for A* would be $f(n) = g(n) + \alpha^*[N - d(n)]$ and, since it is not admissible, the question arises whether C_t, the cost of the solution found by A*, is likely to deviate substantially from the optimal cost C^*. A probabilistic analysis shows that, although in the worst case M may reach a value as high as N, it is most likely to fall in the neighborhood of $\alpha^* N$ or C^*. More precisely, it can be shown (see Appendix B) that, as $N \to \infty$, $P[M \geq (1 + \epsilon)C^*] \to 0$ for every $\epsilon > 0$. Thus, invoking Theorem 2, we can guarantee that, as $N \to \infty$, A* will almost always find a solution path within a $1 + \epsilon$ cost ratio of the optimal, regardless of how small ϵ is.

Theorem 2 can also be used to check for strict admissibility, that is, $C_t = C^*$; all we need to do is to verify the equality $\psi^{-1}(M) = C^*$. This, however, is more conveniently accomplished with the help of the next corollary. It makes direct use of the facts that (1) an upper bound on f along *any* solution path constitutes an upper bound on M and (2) the relation between $f_P(n)$ and C^* is more transparent along an *optimal path*. For example, in the case of A* with $h \leq h^*$, the relation $f_{P^*}(n) \leq C^*$ is self evident.

COROLLARY 1. *If in every graph searched by BF* there exists at least one optimal solution path along which f attains its maximal value on the goal node, then BF* is admissible.*

PROOF. Let BF* terminate with solution path $P_j^S = s, \ldots, t$ and let $P^* = s, \ldots, \gamma$ be an optimal solution path such that $\max_{n \in P^*} f_{P^*}(n) = f_{P^*}(\gamma)$. By Theorem 2, we know that $f_j(t) \leq M$. Moreover, from the definition of M we have $M \leq \max_{n \in P_i^S} f_i(n)$ for every solution path P_i^S. In particular, taking $P_i^S = P^*$, we obtain

$$f_j(t) \leq M \leq \max_{n \in P^*} f_{P^*}(n) = f_{P^*}(\gamma).$$

However, from (1) we know that f is monotonic increasing in C when evaluated on complete solution paths; thus,

$$\psi(C(P_j^S)) \leq \psi(C^*),$$

implying

$$C(P_j^S) \leq C^*,$$

which means that BF* terminates with an optimal-cost path. □

By way of demonstrating its utility, Corollary 1 can readily delineate the range of admissibility of Pohl's weighted evaluation function $f_w = (1 - w)g + wh$, $0 \leq w \leq 1$ (see [18]). Here $\psi(C) = (1 - w)C$, which complies with (1) for $w < 1$. It remains to examine what values of $w < 1$ will force f_w to attain its maximum at the end of some optimal path $P^* = s, \ldots, \gamma$. Writing

$$f_{P^*}(n) \leq f_{P^*}(\gamma),$$

we obtain

$$(1 - w)g^*(n) + wh(n) \leq (1 - w)g^*(\gamma) = (1 - w)[g^*(n) + h^*(n)],$$

or

$$\frac{1 - w}{w} \geq \frac{h(n)}{h^*(n)}.$$

Clearly, if the ratio $h(n)/h^*(n)$ is known to be bounded from above by a constant β, then

$$w < \frac{1}{1 + \beta}$$

constitutes the range of admissibility of BF*. Note that the use of $w > \frac{1}{2}$ may be permissible if h is known to consistently underestimate h^* such that $h(n)/h^*(n) \leq \beta < 1$. Conversely, if h is known to be nonadmissible with $\beta > 1$, then the use of $w = 1/(1 + \beta)$ will turn BF* admissible.

Another useful application of Corollary 1 is to check whether a given combination of g and h, $f = f(g, h)$, would constitute an admissible heuristic in problems of minimizing additive cost measures. If $h \leq h^*$ and f is monotonic in both arguments, then Corollary 1 states that $f(g, h)$ is guaranteed to be admissible as long as

$$f(g, C - g) \leq f(C, 0) \qquad \text{for} \quad 0 \leq g \leq C.$$

Thus, for example, $f = \sqrt{g^2 + h^2}$ is admissible, whereas $f = (g^{1/2} + h^{1/2})^2$ is not. In general, any combination of the form $f = \phi[\phi^{-1}(g) + \phi^{-1}(h)]$ will be admissible if ϕ is monotonic nondecreasing and concave.

3. *Conditions for Node Expansion*

In Section 3.1 we present separate conditions of necessity and sufficiency for nodes expanded by BF* on graphs. Further restricting the problem domain to trees will

enable us to establish, in Section 3.2, an expansion condition that is both necessary and sufficient.

3.1 Expansion Conditions for Graphs

LEMMA 4. *BF* chooses for expansion at least one node n such that at the time of this choice* $f(n) = M$.

PROOF. Let BF* terminate with P_j^S and let $n^* \in P_j^S$ be such that $f_j(n^*) = M_j$. From Lemma 3, $M_j = M$. Moreover, at the time that n^* is last expanded, it is pointed along P_j^S. Hence,

$$f(n^*) = f_j(n^*) = M_j = M. \qquad \square$$

THEOREM 3. *Any node expanded by BF* has* $f(n) \leq M$ *immediately before its expansion.*

PROOF. Follows directly from Lemma 2. $\square$

THEOREM 4. *Let* n^* *be the first node with* $f(n^*) = M$ *that is expanded by BF* (there is at least one). Any node that resides in OPEN with* $f(n) < M$ *prior to the expansion of* n^* *will be selected for expansion before* n^*.

PROOF. $f(n)$ can only decrease through the redirection of its pointers. Therefore, once n satisfies $f(n) < M$, it will continue to satisfy this inequality as long as it is in OPEN. Clearly, then, it should be expanded before n^*. $\square$

Note the difference between Theorems 3 and 4 and their counterparts, results 4 and 5, for A*. First, M plays the role of C^*. Second, the sufficient condition for expansion in Theorem 4, unlike that of result 5, requires that n not merely reside in OPEN but also that it enter OPEN *before* n^* is expanded. For a general f, it is quite possible that a node n may enter OPEN satisfying $f(n) < f(n^*) = M$ and still not be expanded.

We now show that such an event can only occur to *descendants* of nodes n^* for which $f(n^*) = M$; that is, it can only happen to a node n reachable by an M-bounded path, but not by a strictly M-bounded path.

Definition. A path P is called *M-bounded* if every node n ($n \neq s$) along P satisfies $f_P(n) \leq M$. Similarly, if a strict inequality holds for every n along P, we say that P is *strictly M-bounded.*

THEOREM 5. *Any node reachable from s by a strictly M-bounded path will be expanded by BF*.*

PROOF. Consider a strictly M-bounded path P from s to n. We can prove by induction from s to n that every node along P enters OPEN before n^* is expanded and hence, using Theorem 4, that n will be expanded before n^*. $\square$

In Section 4, we use this result to compare the performance of A* to that of other (generalized) best-first algorithms.

The final results we wish to establish now are necessary and sufficient conditions for node expansion that are superior to Theorems 4 and 5 in that they also determine the fate of the descendants of n^*.

THEOREM 6. *Let* P_j^S *be the solution path eventually found by BF* and let* n_i *be the depth-i node along* P_j^s, $i = 0, 1, \ldots$. *A necessary condition for expanding an arbitrary node n in the graph is that, for some* $n_i \in P_j^s$, *there exists an* L_i*-bounded*

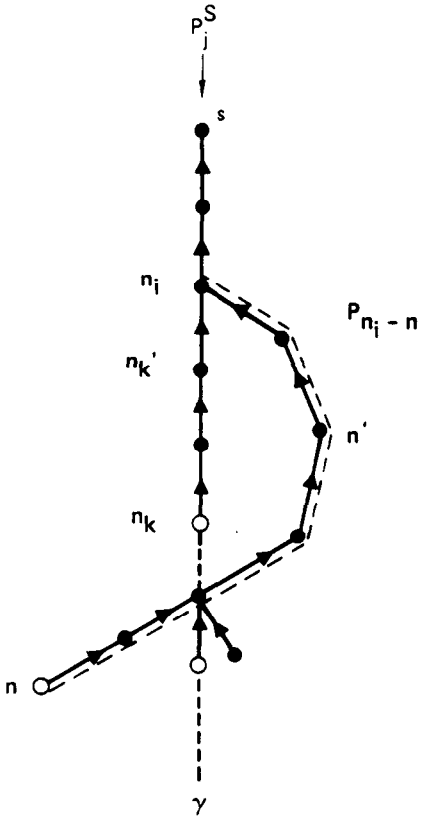


FIG. 1. The condition for expanding node n given that path P_j^s is eventually found by the search. n will be expanded if all the f values along path P_{n_i-n} are lower than the maximal f value between n_i and γ (along P_j^s).

path from n_i to n where $L_i = max_{k>i} f_j(n_k)$. In other words, there should exist a path P_{n_i-n} along which

$$f(n') \leq \max_{k>i} f_j(n_k) \qquad \forall n' \in P_{n_i-n}. \tag{3}$$

Moreover, a sufficient condition for expanding n is that (3) *be satisfied with strict inequality.*

PROOF. Assume that n is expanded by BF* and let n_k be the shallowest OPEN node on P_j^S at time t_n when n is selected for expansion (see Figure 1).

Since P_j^S is the solution path eventually found by BF*, we know (see proof of Lemma 3) that at time $t_n n_k$ is pointed *along* P_j^S and, therefore,

$$f(n) \leq f(n_k) = f_j(n_k).$$

We are now ready to identify the node n_i on P_j^S, which satisfies (3). Let P_{s-n} be the path along which n's pointers are directed at time t_n, and let n_i be the deepest common ancestor of n and n_k along their respective pointer paths P_{s-n} and P_j^S. Since n_i is an ancestor of n_k we have $i < k$ and so $f(n) \leq f_j(n_k)$ implies

$$f(n) \leq \max_{k>i} f_j(n_k).$$

We now repeat this argument for every ancestor n' of n along the P_{n_i-n} segment of P_{s-n}. At the time it was last expanded, each such ancestor n' may have encountered a different $n_{k'}$ in OPEN, but each such $n_{k'}$ must have been a descendent

of n_i along P_j^S satisfying $f(n') \leq f_j(n_{k'})$. Hence, (3) must be satisfied for all nodes n' along the P_{n_i-n} segment of P_{s-n}, which proves the necessary part of Theorem 6.

Sufficiency is proved by assuming that $\max_{k>i} f_j(n_k)$ occurs at some node $n_{k'} \in P_j^S$, $k' > i$. Both n_i and $n_{k'}$ are eventually expanded by BF* and so, if n is not already expanded at the time t' when $n_{k'}$ is last selected for expansion, then P_{n_i-n} should contain at least one OPEN node. We now identify n' as the shallowest OPEN node on P_{n_i-n} at time t', for which we know that

$$f(n') \leq f_{P_{n_i-n}}(n').$$

However, since (3) is assumed to hold with strict inequality for any n' along P_{n_i-n}, we must conclude that

$$f(n') \leq f_{P_{n_i-n}}(n') < f(n_k),$$

implying that n', not n_k, should have been chosen for expansion at time t', thus contradicting our supposition that n remains unexpanded at time t'. □

The expansion condition of Theorem 6 plays a major role in analyzing the average complexity of nonadmissible algorithms, where f is treated as a random variable [16]. This condition is rather complex for general graphs since many paths may stem from P_j^s toward a given node n, all of which must be tested according to Theorem 6. The test is simplified somewhat in the case of trees since (3) need only be tested for one node, $n_i \in P_j^s$, which is the deepest common ancestor of n and γ. Still, the condition stated in Theorem 6 requires that we know which path will eventually be found by the algorithm, and this, in general, may be a hard task to determine a priori. An alternative condition, involving only the behavior of f across the tree, is given in the next subsection.

3.2 A Necessary and Sufficient Condition for Node Expansion on Trees. Here we present a necessary and sufficient condition for node expansion by algorithm BF* under the assumption that the graph to be searched is a tree and the tie-breaking rule is "leftmost-first." This condition, unlike those in Section 3.1, does not require knowing the solution path found but invokes only the properties of the search graph and the evaluation function f.

The following notation and definitions are used:

- T_s A tree rooted at node s. The children of each node are ordered from left to right. N denotes the depth of T_s. Some of its leaf nodes are marked as goal nodes.
- T_r A subtree of T_s rooted at node r, $r \in T_s$.
- M_r The minmax value M associated with solution paths in subtree T_r. If T_r contains no goal nodes, we define $M_r = \infty$.
- P_{i-r} The path from i to r (including, as usual, the two end nodes).
- $P_{(i-r]}$ The path from i to r excluding node i.
- $P_{[i-r)}$ The path from i to r excluding node r.
- $P_{(i-r)}$ The path from i to r excluding nodes i and r.
- $A_{i,j}$ The deepest common ancestor of nodes i and j for which i is not a descendent of j and j is not a descendent of i.
- $f(r)$ The evaluation function $f_{P_{s-r}}(r)$.

For any ancestor node n of a given node k, we now define two parameters called *left value* and *right value* to be the minimum among the M values of the left and

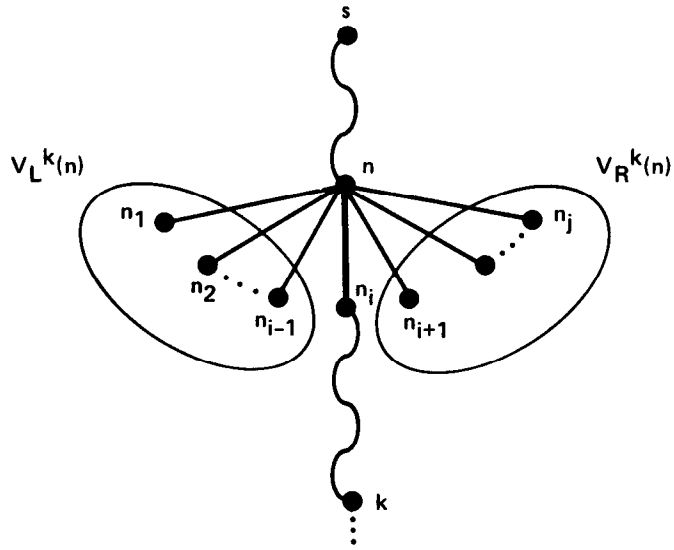


FIG. 2. The left and right values of ancestor n of k, with respect to the path P_{s-k}.

right subtrees rooted at n, respectively. The terms left and right are defined with respect to path P_{s-k} (see Figure 2).

Definition. Let k be a node at depth k ($k \leq N$) in T_s and let n be a node on $P_{[s-k)}$, with its sons $n_1, n_2, \ldots, n_i, \ldots, n_b$ ordered from left to right where n_i is on path P_{s-k}. The *left value of n with respect to k*, $V_L^k(n)$, is given by

$$V_L^k(n) = \min_{1 \leq j \leq i-1} \{M_{n_j}\}.$$

Similarly, the *right value of n with respect to k* is

$$V_R^k(n) = \min_{i+1 \leq j \leq b} \{M_{n_j}\}.$$

Definition. We say that n_i *is to the left of* n_j if some ancestor of n_i is a left sibling of an ancestor of n_j.

Obviously, for any two nodes in T_s either one of them is a *descendant* of the other or one of them is to the *left* of the other (but not both). This renders the execution of algorithm BF* on T_s unique, since, at any time, all nodes in OPEN are totally ordered by both the size of f and the "left of" relationship.

LEMMA 5. *Let k be a node in T_s that is expanded by BF* and let n be to the left of k with $n' = A_{n,k}$. If the path $P_{n'-n}$ is bounded above by $f(k)$, then n will be expanded before k (see Figure 3a).*

PROOF. Assume $P_{n'-n}$ is bounded above by $f(k)$ but k is expanded before n. Immediately before k is expanded, there is a node n'' on $P_{n'-n}$ that is on OPEN. Since we have $f(n'') \leq f(k)$ and n'' is to the left of k (since n is), n'' should be selected for expansion and not k, which contradicts our supposition. □

We are now ready to present the condition for node expansion.

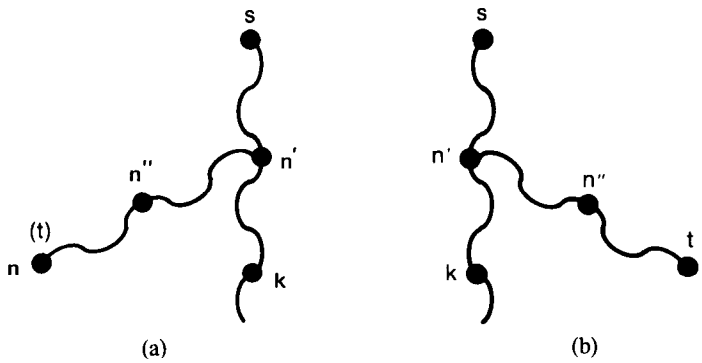


FIG. 3. Notation used is the proof of Lemma 5; if $P_{n'-n}$ is $f(k)$-bounded, then n will be expanded before k. (a) Case (a) of the proof. (b) Case (b) of the proof.

THEOREM 7. *A node k at depth k of the search tree T_s will be expanded by BF* if and only if the f value for each ancestor n of k along the path P_{s-k}, is lower than the minimum M values of all subtrees stemming to the left of the path P_{s-n}, and is not higher than the minimum M values of all subtrees stemming to the right of P_{s-n}. Formally:*

For every n on P_{s-k},

$$f(n) < min\{V_L^k(n') \mid n' \in P_{[s-n)}\}$$

and

$$f(n) \leq min\{V_R^k(n') \mid n' \in P_{[s-n)}\}.$$

PROOF. In part (1), we show that the condition is necessary and in part (2) we prove its sufficiency.

(1) Assume to the contrary that node k was expanded but there is a node on $P_{(s-k)}$ that does not satisfy the corresponding inequality. Let n be the first such node with this counterproperty; that is, either

(a) $f(n) \geq \min\{V_L^k(n') \mid n' \in P_{(s-n)}\}$, or
(b) $f(n) > \min\{V_R^k(n') \mid n' \in P_{(s-n)}\}$.

Assume that (a) holds and let n^* be a node on $P_{(s-n)}$, on which the minimum of the right-hand side of (a) is attained, that is,

$$V_L^k(n^*) = \min\{V_L^k(n') \mid n' \in P_{(s-n)}\}.$$

According to the definition of $V_L^k(n^*)$, there is a path P_{n^*-t} from n^* to a goal node t that is situated to the left of n which is bounded by $V_L^k(n^*)$. Since $V_L^k(n^*) \leq f(n)$, the path P_{n^*-t} is bounded by $f(n)$ and thus, by Lemma 5, t will be expanded before n and the algorithm will halt without expanding n and k, which contradicts our supposition that k is expanded by BF*.

Assume now that (b) holds. Using the same argument, we should conclude that, exactly before n is chosen for expansion, there is a path from an OPEN node to a goal (situated to the right of n) that is *strictly* bounded below $f(n)$. This path should be expanded before n and the algorithm will terminate without expanding n or k.

(2) Let k be a node at depth k that satisfies the condition. We show that k must be expanded by BF*. Let t be the goal node on which BF* halts. If t is a descendant of k, then, obviously, k must be expanded. Otherwise, t is either to the left of k or to its right.

Case 1. t is to the left of k. Let $n' = A_{t,k}$, n' is on the path P_{s-k}. Let M be the $\max f$ value on $P_{(n'-t]}$ and $n'' \in P_{(n'-t]}$ with $f(n'') = M$ (see Figure 3a). Since t is expanded, n' must be expanded, and, therefore, before the algorithm expands n'', there is always an OPEN node on $P_{(n'-k]}$. From the condition of the theorem,

$$f(n) < M = f(n'') \qquad \forall n \in P_{(n'-k]},$$

and therefore all the nodes on $P_{(n'-k]}$ must be expanded before n''. Knowing that n'' is expanded implies the expansion of k and all its ancestors.

Case 2. t is to the right of k. The situation in this case is shown in Figure 3b. From the condition of the theorem, it follows that the path $P_{(n'-k]}$ is bounded below $f(n'')$ and therefore, from Lemma 5, k should be expanded before n''. □

4. *On the Optimality of A**

4.1 Previous Works and the Notion of Equally Informed Algorithms. The *optimality* of A*, in the sense of *computational efficiency*, has been a subject of some confusion. The well-known property of A*, which predicts that decreasing errors $h^* - h$ can only improve its performance [14, result 6], has often been interpreted to reflect some supremacy of A* over other search algorithms of equal information. Consequently, several authors have assumed that optimality of A* is an established fact (e.g., [2, 12, 13]). In fact, all this property says is that some A* algorithms are better than other A* algorithms, depending on the heuristics that guide them. It does not indicate whether the additive rule $f = g + h$ is the best way of combining g and h; neither does it assure us that expansion policies based only on g and h can do as well as more sophisticated policies that use the entire information gathered by the search. These two conjectures are examined in this section, and are given a qualified confirmation.

The first attempt to prove the optimality of A* was carried out by Hart et al. [6, 7] and is summarized in Nilsson [13]. Basically, Hart et al. argue that if some admissible algorithm B fails to expand a node n expanded by A*, then B must have known that any path to a goal constrained to go through node n is nonoptimal. A*, by comparison, had no way of realizing this fact because when n was chosen for expansion it satisfied $g(n) + h(n) \leq C^*$, clearly advertising its promise to deliver an optimal solution path. Thus, the argument goes, B must have obtained extra information from some external source, unavailable to A* (perhaps by computing a higher value for $h(n)$), and this disqualifies B from being an "equally informed," fair competitor to A*.

The weakness of this argument is that it fails to account for two legitimate ways in which B can decide to refrain from expanding n based on information perfectly accessible to A*. First, B may examine the properties of previously exposed portions of the graph and infer that n actually deserves a much higher estimate than $h(n)$. On the other hand, A*, although it has the same information available to it in CLOSED, cannot put it into use because it is restricted to take the estimate $h(n)$ *at face value* and only judge nodes by the score $g(n) + h(n)$. Second, B may also gather information while exploring sections of the graph unvisited by A*, and this should not render B an unfair, "more informed" competitor to A* because, in

principle, A* too had an opportunity to visit those sections of the graph. Later in this section (see Figure 6), we demonstrate the existence of an algorithm B, which manages to outperform A* using this kind of information.

Gelperin [4] has correctly pointed out that, in any discussion of the optimality of A*, one should also consider algorithms that adjust their h in accordance with the information gathered during the search. His analysis, unfortunately, falls short of considering the entirety of this extended class, having to follow an overrestrictive definition of *equally informed.* Gelperin's interpretation of "an algorithm B [that] is *never more informed* than A*," instead of just restricting B from using information inaccessible to A, actually forbids B from processing common information in a better way than A does. For example, if B is a best-first algorithm guided by f_B, then in order to qualify for Gelperin's definition of "never more informed than A*," B is forbidden from ever assigning to a node n a value $f_B(n)$ higher than $g(n) + h(n)$, even if the information gathered along the path to n justifies such an assignment.

In our analysis, we use the natural definition of "equally informed," allowing the algorithms compared to have access to the same heuristic information while placing no restriction on the way they use it. Accordingly, we assume that an arbitrary heuristic function $h(n)$ is assigned to the nodes of G and that the value $h(n)$ is made available to each algorithm that chooses to generate node n. This amounts to viewing $h(n)$ as part of the parameters that specify problem instances, and, correspondingly, we represent each problem instance by the quadruple $I = (G, s, \Gamma, h)$.

We demand, however, that A* only be compared to algorithms that return optimal solutions in those problem instances where their computational performances are to be appraised. In particular, if our problem space contains only cases in which $h(n) \leq h^*(n)$ for every n in G, we only consider algorithms that, like A*, return least-cost solutions in such cases. The class of algorithms answering this conditional admissibility requirement is simply called *admissible* and is denoted by $\mathbf{A}_{ad}$. From this general class of algorithms, we later examine two subclasses $\mathbf{A}_{gc}$ and $\mathbf{A}_{bf}$. $\mathbf{A}_{gc}$ denotes the class of algorithms that are *globally compatible* with A*, that is, they return optimal solutions whenever A* does, even in cases where $h > h^*$. $\mathbf{A}_{bf}$ stands for the class of admissible BF* algorithms, that is, those that conduct their search in a best-first manner, being guided by a path-dependent evaluation function as in Section 1.

Additionally, we assume that each algorithm compared with A* uses the primitive computational step of *node expansion*, that it only expands nodes that were generated before, and that it begins the expansion process at the start node s. This excludes, for instance, bidirectional searches [19] or algorithms that simultaneously grow search trees from several "seed nodes" across G.

4.2. NOMENCLATURE AND A HIERARCHY OF OPTIMALITY RELATIONS. Our notion of *optimality* is based on the usual requirement of *dominance* [14].

Definition. Algorithm A is said to *dominate* algorithm B relative to a set $\mathbf{I}$ of problem instances iff in every instance $I \in \mathbf{I}$, the set of nodes expanded by A is a subset of the set of nodes expanded by B. A *strictly dominates* B iff A dominates B and B does not dominate A, that is, there is at least one instance when A skips a node that B expands, and no instance when the opposite occurs.

This definition is rather stringent because it requires that A establish its superiority over B under two difficult tests:

(1) expanding a *subset* of nodes rather than a *smaller number* of nodes,
(2) outperforming B in *every* problem instance rather than in the *majority* of instances.

Unfortunately, there is no easy way of loosening any of these requirements without invoking statistical assumptions regarding the relative likelihood of instances in **I**. In the absence of an adequate statistical model, requiring dominance remains the only practical way of guaranteeing that A expands *fewer* nodes than B, because, if in some problem instance we were to allow B to skip even one node that is expanded by A, one could immediately present an infinite set of instances when B *grossly* outperforms A. (This is normally done by appending to the node skipped a variety of trees with negligible costs and very low h.)

Adhering to the concept of dominance, the strong definition of optimality proclaims algorithm A *optimal* over a class **A** of algorithms iff A dominates every member of **A**. Here the combined multiplicity of **A** and **I** also permits weaker definitions; for example, we may proclaim A *weakly optimal* over **A** if no member of **A** strictly dominates A. The spectrum of optimality conditions becomes even richer when we examine A*, which stands for not just one but a whole family of algorithms, each defined by the tie-breaking rule chosen. We chose to classify this spectrum into the following four types (in a hierarchy of decreasing strength):

Type 0. A* is said to be 0-optimal over **A** relative to **I** iff, in every problem instance, $I \in \mathbf{I}$, every tie-breaking rule in A* expands a subset of the nodes expanded by any member of **A**. (In other words, every tie-breaking rule dominates all members of **A**.)

Type 1. A* is said to be 1-optimal over **A** relative to **I** iff, in every problem instance, $I \in \mathbf{I}$, there exists at least one tie-breaking rule that expands a subset of the set of nodes expanded by any member of **A**.

Type 2. A* is said to be 2-optimal over **A** relative to **I** iff there exists no problem instance $I \in \mathbf{I}$ in which some member of **A** expands a proper subset of the set of nodes that are expanded by some tie-breaking rule in A*.

Type 3. A* is said to be 3-optimal over **A** relative to **I** iff the following holds: If there exists a problem instance $I_1 \in \mathbf{I}$ where some algorithm $\mathrm{B} \in \mathbf{A}$ skips a node expanded by some tie-breaking rule in A*, then there must also exist some problem instance $I_2 \in \mathbf{I}$ where that tie-breaking rule skips a node expanded by B. (In other words, no tie-breaking rule in A* is strictly dominated by some member of **A**.)

Type 1 describes the notion of optimality most commonly used in the literature, and it is sometimes called "optimal up to a choice of a tie-breaking rule." Note that these four definitions are applicable to any class of algorithms, **B**, contending to be optimal over **A**; we need only replace the words "tie-breaking rule in A*" by the words "member of **B**." If **B** turns out to be a singleton, then type 0 and type 1 collapse to strong optimality. Type 3 will collapse into type 2 if we insist that I_1 be identical to I_2.

We are now ready to introduce the four domains of problem instances over which the optimality of A* is to be examined. The first two relate to the admissibility and consistency of $h(n)$.

Definition. A heuristic function $h(n)$ is said to be *admissible* on (G, Γ) iff $h(n) \leq h^*(n)$ for every $n \in G$.

Definition. A heuristic function $h(n)$ is said to be *consistent* (or *monotone*) on G iff, for any pair of nodes n' and n, the triangle inequality holds:

$$h(n') \leq k(n', n) + h(n).$$

Corresponding to these two properties, we define the following sets of problem instances:

$$\mathbf{I}_{AD} = \{(G, s, \Gamma, h) \mid h \leq h^* \text{ on } (G, \Gamma)\},$$
$$\mathbf{I}_{CON} = \{(G, s, \Gamma, h) \mid h \text{ is consistent on } G\}.$$

Clearly, consistency implies admissibility [16] but not vice versa; therefore, $\mathbf{I}_{CON} \subseteq \mathbf{I}_{AD}$.

A special and important subset of $\mathbf{I}_{AD}$ (and $\mathbf{I}_{CON}$), called *nonpathological instances*, are those instances for which there exists at least one optimal solution path along which h is not fully informed; that is, $h < h^*$ for every nongoal node on that path. The nonpathological subsets of $\mathbf{I}_{AD}$ and $\mathbf{I}_{CON}$ are denoted by $\mathbf{I}_{\bar{AD}}$ and $\mathbf{I}_{\bar{CON}}$, respectively.

It is known that, if $h \leq h^*$, then A* expands every node reachable from s by a strictly C^*-bounded path, regardless of the tie-breaking rule used. The set of nodes with this property is referred to as *surely expanded* by A*. In general, for an arbitrary constant d and an arbitrary evaluation function f over (G, s, Γ, h), we let $\mathbf{N}_f^d$ denote the *set of all nodes reachable from s by some strictly d-bounded path* in G. For example, $\mathbf{N}_{g+h}^{C^*}$ is a set of nodes surely expanded by A* in some instance of $\mathbf{I}_{AD}$.

The importance of nonpathological instances lies in the fact that in such instances the set of nodes surely expanded by A* are indeed *all* the nodes expanded by it. Therefore, for these instances, any claim regarding the set of nodes surely expanded by A* can be translated to "the set of all the nodes" expanded by A*. This is not the case, however, for pathological instances in $\mathbf{I}_{AD}$; $\mathbf{N}_{g+h}^{C^*}$ is often a proper subset of the set of nodes actually expanded by A*. If h is consistent, then the two sets differ only by nodes for which $h(n) = C^* - g^*(n)$ [16]. However, in cases in which h is inconsistent, the difference may be very substantial; each node n for which $h(n) = C^* - g^*(n)$ may have many descendents assigned lower h values (satisfying $h + g < C^*$) and these descendants may be expanded by every tie-breaking rule of A* even though they do not belong to $\mathbf{N}_{g+h}^{C^*}$.

In Section 4.3 we present several theorems regarding the behavior of competing classes of algorithms relative to the set $\mathbf{N}_{g+h}^{C^*}$ of nodes surely expanded by A*, and we interpret these theorems as claims about the type of optimality that A* enjoys over the competing classes. Moreover, for any given pair (**A**, **I**) where **A** is a class of algorithms drawn from $\{\mathbf{A}_{ad}, \mathbf{A}_{bf}, \mathbf{A}_{gc}\}$ and **I** is a domain of problem instances from $\{\mathbf{I}_{AD}, \mathbf{I}_{\bar{AD}}, \mathbf{I}_{CON}, \mathbf{I}_{\bar{CON}}\}$, we determine the strongest type of optimality that can be established over **A** relative to **I** and identify the algorithm that achieves this optimality. The relationships between these classes of algorithms and problem domains are shown in Figure 4. The algorithm A** is an improvement over A* discussed in Appendix B.

4.3 Where and How Is A* Optimal?

4.3.1 *Optimality over Admissible Algorithms,* $\mathbf{A}_{ad}$

THEOREM 8. *Any algorithm that is admissible on* $\mathbf{I}_{AD}$ *will expand, in every instance* $I \in \mathbf{I}_{CON}$, *all nodes surely expanded by A*.*

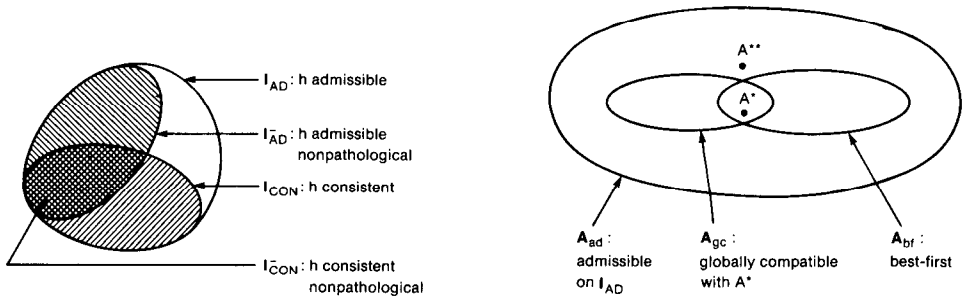


FIG. 4. Venn diagrams showing the classes of algorithms and the domains of problem instances relative to which the optimality of A* is examined. (a) Problem domains. (b) Classes of algorithms.

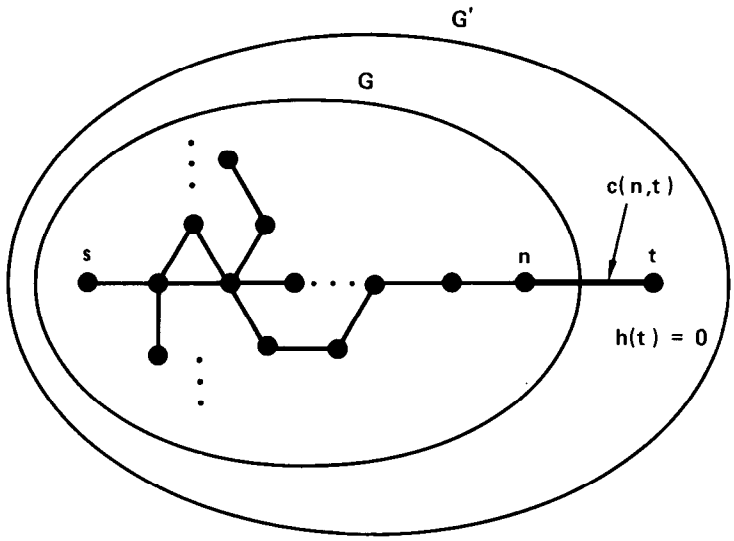


FIG. 5. The graph G' represents a new problem instance constructed by appending to n a branch leading to a new goal node t.

PROOF. Let $I = (G, s, \Gamma, h)$ be some problem instance in $\mathbf{I}_{AD}$ and assume that n is surely expanded by A*, that is, $n \in \mathbf{N}^{C^*}_{g+h}$. Therefore, there exists a path P_{s-n} such that

$$g(n') + h(n') < C^* \qquad \forall n' \in P_{s-n}.$$

Let B be an algorithm compatible with A*, namely, halting with cost C^* in I.

Assume that B does not expand n. We now create a new graph G' (see Figure 5) by adding to G a goal node t with $h(t) = 0$ and an edge from n to t with nonnegative cost $c = h(n) + \Delta$, where

$$\Delta = \tfrac{1}{2}(C^* - D) > 0 \qquad \text{and} \qquad D = \max\{f(n') \mid n' \in \mathbf{N}^{C^*}_{g+h}\}.$$

This construction creates a new solution path P^* with cost at most $C^* - \Delta$ and, simultaneously (owing to h's consistency on I), retains the consistency (and admissibility) of h on the new instance I'. To establish the consistency of h in I', we note that since we kept the h values of all nodes in G unchanged, consistency

will continue to hold between any pair of nodes previously in G. It remains to verify consistency on pairs involving the new goal node t, which amounts to establishing the inequality $h(n') \leq k(n', t)$ for every node n' in G. Now, if at some node n' we have $h(n') > k(n', t)$, then we should also have

$$h(n') > k(n', n) + c = k(n', n) + h(n) + \Delta$$

in violation of h's consistency on I. Thus, the new instance is also in $\mathbf{I}_{\text{CON}}$.

In searching G', algorithm A* will find the extended path P^* costing $C^* - \Delta$, because

$$f(t) = g(n) + c = f(n) + \Delta \leq D + \Delta = C^* - \Delta < C^*,$$

and so, t is reachable from s by a path bounded by $C^* - \Delta$, which ensures its selection. Algorithm B, on the other hand, if it avoids expanding n, must behave the same as in problem instance I, halting with cost C^*, which is higher than that found by A*. This contradicts the supposition that B is both admissible on I and avoids the expansion of node n. □

The implications of Theorem 8 relative to the optimality of A* are rather strong. In nonpathological cases, $I \in \bar{\mathbf{I}}_{\text{CON}}$ A* never expands a node outside $\mathbf{N}_{g+h}^{C^*}$ and, therefore, Theorem 8 establishes the 0-optimality of A* over all admissible algorithms relative to $\bar{\mathbf{I}}_{\text{CON}}$. In pathological cases of $\mathbf{I}_{\text{CON}}$, there may also be nodes satisfying $f(n) = C^*$ that some tie-breaking rule in A* expands, and, since these nodes are defined to be outside $\mathbf{N}_{g+h}^{C^*}$, they may be avoided by some algorithm $\mathrm{B} \in \mathbf{A}_{\text{ad}}$, thus destroying the 0-optimality of A* relative to all $\mathbf{I}_{\text{CON}}$. However, since there is always a tie-breaking rule in A*, which, in addition to $\mathbf{N}_{g+h}^{C^*}$, expands only nodes along one optimal path, Theorem 8 also establishes the 1-optimality of A* relative to the entire $\mathbf{I}_{\text{CON}}$ domain. Stronger yet, the only nodes that A* expands outside $\mathbf{N}_{g+h}^{C^*}$ are those satisfying $f(n) = C^*$, and since this equality is not likely to occur in many nodes of the graph, we may interpret Theorem 8 to endow A* with "almost" 0-optimality (over all admissible algorithms) relative to $\mathbf{I}_{\text{CON}}$.

The proof of Theorem 8 makes it tempting to conjecture that A* retains the same type of optimality relative to cases in which h is admissible but not necessarily consistent. In fact, the original argument of Hart et al. [6], that no admissible algorithm equally informed to A* can ever avoid a node expanded by A* (see Section 4.1), amounts to claiming that A* is at least 1-optimal relative to $\mathbf{I}_{\text{AD}}$. Similar claims are made by Méro [12] and are suggested by the theorems of Gelperin [4].

Unfortunately, Theorem 8 does not lend itself to such extension; if h is admissible but not consistent, then, after adding the extra goal node t to G (as in Figure 5), we can no longer guarantee that h will remain admissible on the new instance created. Furthermore, we can actually construct an algorithm that is admissible on $\mathbf{I}_{\text{AD}}$, and yet, in some problem instances, it will grossly outperform every tie-breaking rule in A*. Consider an algorithm B guided by the following search policy: Conduct an exhaustive right-to-left depth-first search but refrain from expanding one distinguished node n, for example, the leftmost son of s. By the time this search is completed, examine n to see if it has the potential of sprouting a solution path cheaper than all those discovered so far. If it has, expand it and continue the search exhaustively. Otherwise,[3] return the cheapest solution at hand. B is clearly admissible; it cannot miss an optimal path because it will only avoid expanding n when it has sufficient information to justify this action, but otherwise will leave no stone

[3] A simple valid test for skipping a node in I_{AD} is that $\max\{g(n) + h(n) \mid n\}$ on some path P from s to n be larger than the cost of the cheapest solution at hand.

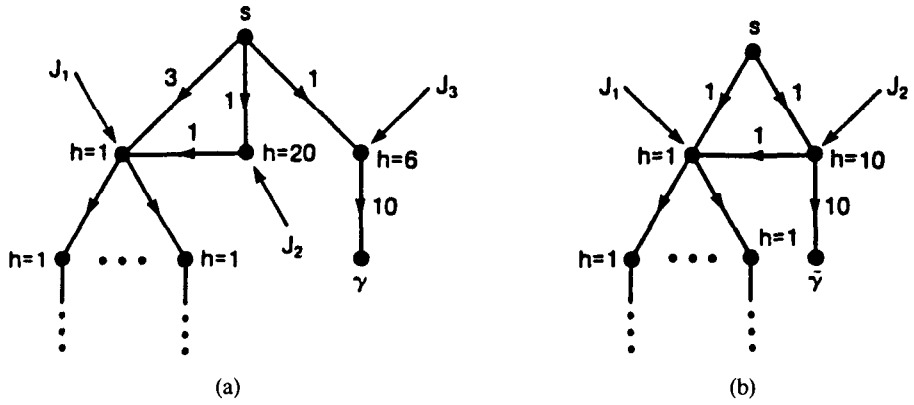


FIG. 6. Graphs demonstrating that A* is not optimal. (a) A nonpathological problem where algorithm B will avoid a node (J_1) that A* must expand. (b) A pathological graph ($f(J_2) = C^* = 11$) where algorithm B expands a proper subset of the nodes expanded by A*.

unturned. Yet, in the graph of Figure 6a, B will avoid expanding many nodes that are surely expanded by A*. A* will expand node J_1 immediately after s because ($f(J_1) = 4$) and subsequently will also expand many nodes in the subtree rooted at J_1. B, on the other hand, will expand J_3, then select for expansion the goal node γ, continue to expand J_2, and, at this point, halt without expanding node J_1. Relying on the admissibility of h, B can infer that the estimate $h(J_1) = 1$ is overly optimistic and should be at least equal to $h(J_2) - 1 = 19$, thus precluding J_1 from lying on a solution path cheaper than the path (s, J_3, γ) at hand.

Granted that A* is not 1-optimal over all admissible algorithms relative to $\mathbf{I}_{AD}$, the question arises if a 1-optimal algorithm exists altogether. Clearly, if a 1-optimal algorithm exists, it would have to be better than A* in the sense of skipping, in some problem instances, at least one node surely expanded by A*, while never expanding a node that is surely skipped by A*. Note that algorithm B could not be such an optimal algorithm because, in return for skipping node J_1 in Figure 6a, it had to pay the price of expanding J_2, and J_2 will not be expanded by A* regardless of the tie breaking rule invoked. If we could show that this "node trade-off" pattern must hold for every admissible algorithm and on every instance of $\mathbf{I}_{AD}$, then we would have to conclude both that no 1-optimal algorithm exists and that A* is 2-optimal relative to this domain. Theorem 9 accomplishes this task relative to $\bar{\mathbf{I}}_{AD}$.

THEOREM 9. *If an admissible algorithm B does not expand a node that is surely expanded by A* in some problem instance when h is admissible and nonpathological, then, in that very problem instance, B must expand a node that is avoided by every tie-breaking rule in A*.*

PROOF. Assume the contrary, that is, that there is an instance $I = (G, s, \Gamma, h) \in \bar{\mathbf{I}}_{AD}$ such that a node n that is surely expanded by A* is avoided by B and, at the same time, B expands no node that is avoided by A*. We show that this assumption implies the existence of another instance $I' \in \mathbf{I}_{AD}$ for which B will not find an optimal solution. I' is constructed by taking the graph G_e exposed by the run of A* (including nodes in OPEN) and appending to it another edge (n, t) to a new goal node t, with cost $c(n, t) = D - k_e(s, n)$ where

$$D = \max\{f(n') \mid n' \in \mathbf{N}^{C^*}_{g+h}\},$$

and $k_e(n', n)$ is the cost of the cheapest path from n' to n in G_e.

Since G contains an optimal path $P^*_{s-\gamma}$ along which $h(n') < h^*(n')$ (with the exception of γ and possibly s), we know that, because ties are broken in favor of goal nodes, A* will halt without ever expanding a node having $f(n) = C^*$. Therefore, every nonterminal node in G_e must satisfy the strict inequality $g(n) + h(n) < C^*$.

We shall first prove that I' is in $\mathbf{I}_{AD}$, that is, that $h(n') \leq h^*_{I'}(n')$ for every node n' in G_e. This inequality certainly holds for n' such that $g(n') + h(n') \geq C^*$ because all such nodes were left unexpanded by A* and hence appear as terminal nodes in G_e, for which $h^*_{I'}(n') = \infty$ (with the exception of γ, for which $h(\gamma) = h^*_{I'}(\gamma) = 0$). It remains, therefore, to verify the inequality for nodes n' in $\mathbf{N}^{C^*}_{g+h}$ for which we have $g(n') + h(n') \leq D$. Assume the contrary, that for some $n' \in \mathbf{N}^{C^*}_{g+h}$ we have $h(n') > h^*_{I'}(n')$. This implies

$$\begin{aligned} h(n') &> k_e(n', n) + c(n, t) \\ &= k_e(n', n) + D - k_e(s, n) \\ &\geq k_e(n', n) + k_e(s, n') + h(n') - k_e(s, n), \end{aligned}$$

or

$$k_e(s, n) > k_e(n', n) + k_e(s, n'),$$

in violation of the triangle inequality for cheapest paths in G_e. Hence, I' is in $\mathbf{I}_{AD}$.

Assume now that algorithm B does not generate any node outside G_e. If B has avoided expanding n in I, it should also avoid expanding n in I'; all decisions must be the same in both cases since the sequence of nodes generated (including those in OPEN) is the same. On the other hand, the cheapest path in I' now goes from s to n to t', having the cost $D < C^*$, and will be missed by B. This violates the admissibility of B on an instance in $\mathbf{I}_{AD}$ and proves that B could not possibly avoid the expansion of n without generating at least one node outside G_e. Hence, B must expand at least one node avoided by A* in this specific run. □

Theorem 9 has two implications. On one hand, it conveys the discomforting fact that neither A* nor any other algorithm is 1-optimal over those guaranteed to find an optimal solution when given $h \leq h^*$. On the other hand, Theorem 9 endows A* with some optimality property, albeit weaker than hoped; the only way to gain one node from A* is to relinquish another. Not every algorithm enjoys such strength. These implications are summarized in the following corollary.

COROLLARY 2. *No algorithm can be 1-optimal over all admissible algorithms relative to $\mathbf{I}_{AD}$, but A* is 2-optimal over this class relative to $\mathbf{I}_{AD}$.*

The fact that Theorem 9 had to be limited to nonpathological instances is explained by Figure 6b, showing an exception to the node-trade-off rule on a pathological instance. Algorithm B does not expand a node (J_1) that must be expanded by A*, and yet B does not expand any node that A* may skip. This example implies that A* is not 2-optimal relative to the entire $\mathbf{I}_{AD}$ domain and, again, this begs the questions whether there exists a 2-optimal algorithm altogether, and whether A* is at least 3-optimal relative to this domain.

The answer to both questions is, again, negative; another algorithm that we call A**, turns out both to strictly dominate A* and to meet the requirements for type 3 optimality relative to $\mathbf{I}_{AD}$. A** conducts the search in a manner similar to A*, with one exception; instead of $f(n) = g(n) + h(n)$, A** uses the evaluation function

$$f(n) = \max\{g(n') + h(n') \mid n' \text{ on the current path to } n\}.$$

This, in effect, is equivalent to raising the value of $h(n)$ to a level where it becomes consistent with the h's assigned to the ancestors of n (see [12]). A**

chooses for expansion the nodes with the lowest f value in OPEN (breaking ties arbitrarily, but in favor of goal nodes) and adjusts pointers along the path having the lowest g value. In Figure 6a, for example, if A** ever expands node J_2, then its son J_1 will immediately be assigned the value $f(J_1) = 21$ and its pointer directed toward J_2.

It is possible to show (see Appendix B) that A** is admissible and that in nonpathological cases A** expands the same set of nodes as does A*, namely, the surely expanded nodes in $\mathbf{N}^{C^*}_{g+h}$. In pathological cases, however, there exist tie-breaking rules in A** that strictly dominate every tie-breaking rule in A*. This immediately precludes A* from being 3-optimal relative to $\mathbf{I}_{AD}$ and nominates A** for that title.

THEOREM 10. *Let a** be some tie-breaking rule in A** and B an arbitrary algorithm, admissible relative to* $\mathbf{I}_{AD}$. *If in some problem instance* $I_1 \in \mathbf{I}_{AD}$, *B skips a node expanded by a**, then there exists another instance* $I_2 \in \mathbf{I}_{AD}$ *where B expands a node skipped by a**.*

PROOF. Let

$$S_A = n_1, n_2, \ldots, n_k, J, \ldots$$

and

$$S_B = n_1, n_2, \ldots, n_k, K, \ldots$$

be the sequences of nodes expanded by a** and B, respectively, in problem instance $I_1 \in \mathbf{I}_{AD}$, that is, K is the first node in which the sequence S_B deviates from S_A. Consider G_e, the explored portion of the graph just before a** expands node J. That same graph is also exposed by B before it decides to expand K instead. Now construct a new problem instance I_2 consisting of G_e appended by a branch (J, t) with cost $c(J, t) = f(J) - g(J)$, where t is a goal node and $f(J)$ and $g(J)$ are the values that a** computed for J before its expansion. I_2 is also in $\mathbf{I}_{AD}$ because $h(t) = 0$ and $c(J, t)$ are consistent with $h(J)$ and with the h's of all ancestors of J in G_e. For, if some ancestor n_i of J satisfies $h(n_i) > h^*(n_i)$, we obtain a contradiction

$$\begin{aligned} g(n_i) + h(n_i) &> g(n_i) + h^*(n_i) \\ &= g(n_i) + k_e(n_i, J) + c(J, t) \\ &\geq g(J) + c(J, t) \\ &= f(J) \equiv \max_j [g(n_j) + h(n_j)]. \end{aligned}$$

Moreover, a** will expand in I_2 the same sequence of nodes as it did in I_1, until J is expanded, at which time t enters OPEN with $f(t) = \max[g(J) + c(J, t), f(J)] = f(J)$. Now, since J was chosen for expansion by virtue of its lowest f value in OPEN, and since a** always breaks up ties in favor of a goal node, the next and final node that a** expands must be t. Now consider B. The sequence of nodes it expands in I_2 is identical to that traced in I_1 because, by avoiding node J, B has no way of knowing that a goal node has been appended to G_e. Thus, B will expand K (and perhaps more nodes on OPEN), a node skipped by a**. □

Note that the special evaluation function used by A** $f(n) = \max\{g(n') + h(n') \mid n' \text{ on } P_{s-n}\}$ was necessary to ensure that the new instance, I_2, remains in $\mathbf{I}_{AD}$. The proof cannot be carried out for A* because the evaluation function $f(n) = g(n) + h(n)$ results in $c(J, t) = h(J)$, which may lead to violation of $h(n_i) \leq h^*(n_i)$ for some ancestor of J.

Theorem 10, together with the fact that its proof makes no use of the assumption that B is admissible, gives rise to the following conclusion.

COROLLARY 3. *A** is 3-optimal over all algorithms relative to* $\mathbf{I}_{AD}$.

Theorem 10 also implies that there is no 2-optimal algorithm over $\mathbf{A}_{ad}$ relative to $\mathbf{I}_{AD}$. From the 3-optimality of A** we conclude that every 2-optimal algorithm, if such exists, must be a member of the A** family, but Figure 6b demonstates an instance of $\mathbf{I}_{AD}$ where another algorithm (B) only expands a proper subset of the nodes expanded by every member of A**. This establishes the following desired conclusion.

COROLLARY 4. *There is no 2-optimal algorithm over* $\mathbf{A}_{ad}$ *relative to* $\mathbf{I}_{AD}$.

4.3.2 *Optimality over Globally Compatible Algorithms,* $\mathbf{A}_{gc}$. So far, our analysis was restricted to algorithms in $\mathbf{A}_{ad}$, that is, those that return optimal solutions if $h(n) \leq h^*(n)$ for all n, but that may return arbitrarily poor solutions if there are some n for which $h(n) > h^*(n)$. In situations in which the solution costs are crucial and in which h may occasionally overestimate h^*, it is important to limit the choice of algorithms to those which return reasonably good solutions even when $h > h^*$. A*, for example, provides such guarantees; the costs of the solutions returned by A* do not exceed $C^* + \Delta$ where Δ is the highest error $h^*(n) - h(n)$ over all nodes in the graph [5], and, moreover, A* still returns optimal solutions in many problem instances, that is, whenever Δ is zero along some optimal path. This motivates the definition of $\mathbf{A}_{gc}$, the class of algorithms *globally compatible* with A*; namely, they return optimal solutions in *every* problem instance in which A* returns such solution.

Since $\mathbf{A}_{gc}$ is a subset of $\mathbf{A}_{ad}$, we should expect A* to hold a stronger optimality status over $\mathbf{A}_{gc}$, at least relative to instances drawn from $\mathbf{I}_{AD}$. The following theorem confirms this expectation.

THEOREM 11. *In problem instances in which h is admissible, any algorithm that is globally compatible with A* will expand all nodes surely expanded by A*.*

PROOF. Let $I = (G, s, \Gamma, h)$ be some problem instance in $\mathbf{I}_{AD}$ and let node n be surely expanded by A*; that is, there exists a path P_{s-n} such that

$$g(n) + h(n) < C^* \qquad \forall n' \in P_{s-n}.$$

Let $D = \max\{f(n') \mid n' \in P_{s-n}\}$ and assume that some algorithm $B \in \mathbf{A}_{gc}$ fails to expand n. Since $I \in \mathbf{I}_{AD}$, both A* and B will return cost C^*, while $D < C^*$.

We now create a new graph G', as in Figure 5, by adding to G a goal node t' with $h(t') = 0$ and an edge from n to t' with nonnegative cost $D - g(P_{s-n})$. Denote the extended path $P_{s-n-t'}$ by P^*, and let $I' = (G', s, \Gamma \cup t', h)$ be a new instance in the algorithms' domain. Although h may no longer be admissible on I', the construction of I' guarantees that $f(n') \leq D$ if $n' \in P^*$, and thus, by Theorem 2, algorithm A* searching G' will find an optimal solution path with cost $C_t \leq M \leq D$. Algorithm B, however, will search I' in exactly the same way it searched I; the only way B can reveal any difference between I and I' is by expanding n. Since it did not, it will not find solution path P^*, but will halt with cost $C^* > D$, the same cost it found for I and worse than that found by A*. This contradicts its property of being globally compatible with A*. □

COROLLARY 5. *A* is 0-optimal over* $\mathbf{A}_{gc}$ *relative to* $\mathbf{I}_{AD}$.

The corollary follows from the fact that in nonpathological instances A* expands *only* surely expanded nodes.

COROLLARY 6. *A* is 1-optimal over* $\mathbf{A}_{gc}$ *relative to* $\mathbf{I}_{AD}$.

PROOF. The proof relies on the observation that, for every optimal path P^* (in any instance of $\mathbf{I}_{AD}$), there is a tie-breaking rule of A* that expands only nodes along P^* plus perhaps some other nodes having $g(n) + h(n) < C^*$; that is, the only nodes expanded satisfying the equality $g(n) + h(n) = C^*$ are those on P^*. Now, if A* is not 1-optimal over $\mathbf{A}_{gc}$, then, given an instance I, there exists an algorithm $B \in \mathbf{A}_{gc}$ such that B avoids some node expanded by all tie-breaking rules in A*. To contradict this supposition, let A_I^* be the tie-breaking rule of A* that returns the same optimal path P^* as B returns, but expands no node outside P^* for which $g(n) + h(n) = C^*$. Clearly, any node n that B avoids and A_I^* expands must satisfy $g(n) + h(n) < C^*$. We can now apply the argument used in the proof of Theorem 11, appending to n a branch to a goal node t', with cost $c(n, t') = h(n)$. Clearly, A_I^* will find the optimal path (s, n, t') costing $g(n) + h(n) < C^*$, whereas B will find the old path costing C^*, thus violating its global compatibility with A*. □

4.3.3 *Optimality over Best-First Algorithms,* $\mathbf{A}_{bf}$. The next result establishes the optimality of A* over the set of best-first algorithms (BF*), which are admissible if provided with $h \le h^*$. These algorithms are permitted to employ any evaluation function f_P, where f is a function of the nodes, the edge-costs, and the heuristic function h evaluated on the nodes of P; that is,

$$f_P \triangleq f(s, n_1, n_2, \ldots, n) = f(\{n_i\}, \{c(n_i, n_{i+1})\}, \{h(n_i)\} \mid n_i \in P).$$

LEMMA 6. *Let B be a BF* algorithm using an evaluation function* f_B *such that for every* $(G, s, \Gamma, h) \in \mathbf{I}_{AD}$ f_B *satisfies*

$$f_{P_i^\gamma} = f(s, n_1, n_2, \ldots, \gamma) = C(P_i^\gamma) \qquad \forall \gamma \in \Gamma.$$

If B is admissible on $\mathbf{I}_{AD}$, *then* $\mathbf{N}_{g+h}^{C^*} \subseteq \mathbf{N}_{f_B}^{C^*}$.

PROOF. Let $I = (G, s, \Gamma, h) \in I_{AD}$ and assume $n \in \mathbf{N}_{g+h}^{C^*}$ but $n \notin \mathbf{N}_{f_B}^{C^*}$; that is, there exists a path P_{s-n} such that for every $n' \in P_{s-n}$ $g_P(n') + h(n') < C^*$ and, for some $n' \in P_{s-n}$, $f_B(n') \ge C^*$.

Let

$$Q = \max_{n' \in Ps-n} \{g(n') + h(n')\},$$
$$Q_B = \max_{n' \in Ps-n} \{f_B(n')\}.$$

Obviously, $Q < C^*$ and $Q_B \ge C^* \Rightarrow Q_B > Q$. Define G' to include path P_{s-n} with two additional goal nodes t_1, t_2 as described by Figure 7. The cost on edge (s, t_2) is $(Q_B + Q)/2$; the cost on edge (n, t_1) is $Q - g_{P_{s-n}}(n)$; t_1 and t_2 are assigned $h' = 0$, while all other nodes retain their old h. $I' = (G', s, \Gamma \cup \{t_1, t_2\}, h) \in \mathbf{I}_{AD}$ since $\forall n'$, $n' \in P_{s-n}$, $g(n') + h(n') \le Q$, which implies that $h(n') \le Q - g(n') = h_{I'}^*(n')$.

Obviously the optimal path in G' is P_{s-t_1} with cost Q. However, following the condition of the lemma, the evaluation function f_B satisfies

$$M_{P_{s-t_2}} = f_B(t_2) = C(P_{s-t_2}) = \frac{Q_B + Q}{2} < Q_B.$$

Moreover, since $M_{P_{s-t_1}} \ge Q_B$, we have $M_{P_{s-t_2}} < M_{P_{s-t_1}}$, implying that B halts on the suboptimal path P_{s-t_2}, thus contradicting its admissibility. □

THEOREM 12. *Let B be a BF* algorithm such that* f_B *satisfies the property of Lemma* 6.

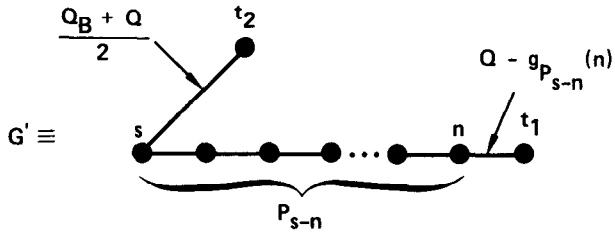
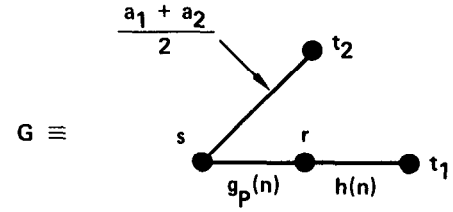


FIG. 7. A graph showing that any admissible best-first algorithm must assign $f_B(n') < C^*$ to every node n' along P_{s-n}.

FIG. 8. A graph demonstrating that an admissible best-first algorithm cannot assign to any node a value $f(n) = F[g(n), h(n)]$ greater than $g(n) + h(n)$.



(*a*) *If B is admissible over* $\mathbf{I}_{AD}$, *then B expands every node in* $\mathbf{N}^{C^*}_{g+h}$.
(*b*) *If B is admissible over* $\mathbf{I}_{AD}$ *and* f_B *is of the form*

$$f_{P_{s-n}}(n) = F(g_{P_{s-n}}(n), h(n)),$$

then $F(x, y) \leq x + y$.

PROOF

(a) Let M be the minmax value corresponding to the evaluation function f_B. It is easy to see that $M \geq C^*$. From that and from Theorem 11 we get

$$\mathbf{N}^{C^*}_{g+h} \subseteq \mathbf{N}^{C^*}_{f_B} \subseteq \mathbf{N}^{M}_{f_B},$$

and it is implied by Theorem 5 that any node in $\mathbf{N}^{M}_{f_B}$ is expanded by B.

(b) Assume to the contrary that there is a path P and a node $n \in P$ such that

$$F(g_P(n), h(n)) > g_P(n) + h(n).$$

Let $a_1 = F(g_P(n), h(n))$, $a_2 = g_P(n) + h(n)$. Obviously, $a_2 < (a_1 + a_2)/2 < a_1$. Let G be a graph like that shown in Figure 8 having nodes s, r, t_1, t_2 and edges (s, r), (r, t_1), (s, t_2), with costs $c(s, r) = g_P(n)$, $c(r, t_1) = h(n)$, $c(s, t_2) = (a_1 + a_2)/2$. Let $I = (G, s, \{t_1, t_2\}, h)$, where $h(s) = 0$, $h(r) = h(n)$, and $h(t_1) = h(t_2) = 0$. Obviously $I \in \mathbf{I}_{AD}$. However,

$$f(r) = F(g_P(n), h(n)) = a_1 > \frac{a_1 + a_2}{2} = c(s, t_2) = f(t_2),$$

implying that B halts on solution path P_{s-t_2}, again contradicting its admissibility. □

COROLLARY 7. *A* is 0-optimal over* $\mathbf{A}_{bf}$ *relative to* $\mathbf{I}^{-}_{AD}$ *and 1-optimal relative to* $\mathbf{I}_{CON}$. *A** is 1-optimal over* $\mathbf{A}_{bf}$ *relative to* $\mathbf{I}_{AD}$.

Note that A** is not a member of $\mathbf{A}_{bf}$; it directs pointers toward the lowest g rather than the lowest f path as instructed by our definition of BF* (Section 1). □

An interesting implication of Part b of Theorem 12 asserts that any admissible combination of g and h, $h \leq h^*$, will expand every node surely expanded by A*.

Class of Algorithms

Domain of Problem Instances	Admissible if $h \leq h^*$ $\mathbf{A}_{ad}$	Globally Compatible with A* $\mathbf{A}_{gc}$	Best-First $\mathbf{A}_{bf}$
Admissible $\mathbf{I}_{AD}$	A** is 3-optimal No 2-optimal exists	A* is 1-optimal No 0-optimal exists	A* is 1-optimal No 0-optimal exists
Admissible and nonpathological $\mathbf{I}_{\bar{AD}}$	A* is 2-optimal No 1-optimal exists	A* is 0-optimal	A* is 0-optimal
Consistent $\mathbf{I}_{CON}$	A* is 1-optimal No 0-optimal exists	A* is 1-optimal No 0-optimal exists	A* is 1-optimal No 0-optimal exists
Consistent nonpathological $\mathbf{I}_{\bar{CON}}$	A* is 0-optimal	A* is 0-optimal	A* is 0-optimal

FIGURE 9

In other words, the additive combination $g + h$ is, in this sense, the optimal way of aggregating g and h for additive cost measures.

The 0-optimality of A* relative to nonpathological instances of $\mathbf{I}_{AD}$ also implies that in these instances $g(n)$ constitutes a sufficient summary of the information gathered along the path from s to n. Any additional information regarding the heuristics assigned to the ancestors of n, or the costs of the individual arcs along the path, is superfluous and cannot yield a further reduction in the number of nodes expanded. Such information, however, may help reduce the number of node *evaluations* performed by the search (see [1, 11, 12]). □

4.4 SUMMARY AND DISCUSSION. Our results concerning the optimality of A* are summarized in Figure 9. For each class-domain combination from Figure 4, the table identifies the strongest type of optimality that exists and the algorithm achieving it.

The most significant results are those represented in the leftmost column, relating to $\mathbf{A}_{ad}$, the entire class of algorithms that are admissible whenever provided with optimistic advice. Contrary to prevailing beliefs, A* turns out not to be optimal

over $\mathbf{A}_{ad}$ relative to every problem graph quantified with optimistic estimates. In some graphs, there are admissible algorithms that will find the optimal solution in just a few steps, whereas A* (as well as A** and all their variations) would be forced to explore arbitrary large regions of the search graphs (see Figure 6a). In bold defiance of the argument of Hart et al. [6] for the optimality of A*, these algorithms succeed in outsmarting A* by penetrating regions of the graph that A* finds unpromising (at least temporarily), visiting some goal nodes in these regions, then processing the information gathered to identify and purge those nodes on OPEN that no longer promise to sprout a solution better than the cheapest one at hand.

In nonpathological cases, however, these algorithms cannot outsmart A* without paying a price. The 2-optimality of A* relative to $\mathbf{I}_{AD}$ means that each such algorithm must always expand at least one node that A* will skip. This means that the only regions of the graph capable of providing node-purging information are regions that A* will not visit *at all*. In other words, A* makes full use of the information gathered along its search, and there could be no gain in changing the order of visiting nodes that A* plans to visit anyhow.

This instance-by-instance node trade-off no longer holds when pathological cases are introduced. The fact that A* is not 2-optimal relative to $\mathbf{I}_{AD}$ means that some smart algorithms may outperform A* by simply penetrating certain regions of the graph *earlier than A** (A* will later visit these regions), thus expanding only a proper subset of the set of nodes expanded by A*. In fact the lack of 2-optimality in the ($\mathbf{A}_{ad}$, $\mathbf{I}_{AD}$) entry of Figure 9 means that no algorithm can be protected against such smart competitors. For any admissible algorithm A_1, there exists another admissible algorithm A_2 and a graph G quantified by optimistic heuristic $h(h \leq h^*)$ such that A_2 expands fewer nodes than A_1 when applied to G. Méro [12] has recently shown that no optimal algorithm exists if complexity is measured by the number of expansion operations (a node can be reopened several times). Our result now shows that $\mathbf{A}_{ad}$ remains devoid of an optimal algorithm even if we measure complexity by the number of distinct nodes expanded.

The type-3 optimality of A** over $\mathbf{A}_{ad}$ further demonstrates that those "smart" algorithms that prevent A* from achieving optimality are not smart after all, but simply lucky; each takes advantage of the peculiarity of the graph for which it was contrived, and none can maintain this superiority over all problem instances. If it wins on one graph, there must be another where it is beaten, and by the very same opponent, A**. It is in this sense that A** is 3-optimal; it exhibits a universal robustness against all its challengers.

Perhaps the strongest claim that Figure 9 makes in favor of A* is contained in the entries related to $\mathbf{I}_{CON}$, the domain of problems in which h is known to be not only admissible, but also consistent. It is this domain that enables A* to unleash its full pruning powers, achieving a node-by-node superiority (types 0 and 1) over all admissible algorithms. Recalling also that, under consistent h, A* never reopens closed nodes and that only few nodes are affected by the choice of tie-breaking rule (see [16]), we conclude that in this domain A* constitutes the most effective scheme of utilizing the advice provided by h.

This optimality is especially significant in light of the fact that consistency is not an exception but rather a common occurrence; almost *all* admissible heuristics invented by people are consistent. The reason is that the technique people invoke in generating heuristic estimates is often that of *relaxation*: We imagine a simplified version of the problem at hand by relaxing some of its constraints and solve the relaxed version mentally; then we use the cost of the resulting solution as a heuristic

for the original problem [15]. It can be shown that any heuristic generated by such a process is automatically consistent, which explains the abundance of consistent cases among human-generated heuristics. Thus, the strong optimality of A* under the guidance of consistent heuristics implies, in effect, its optimality in most cases of practical interest.

Appendix A. Finding an ϵ-Optimal Path in a Tree with Random Costs

Let $C^*(N, p)$ be the optimal cost of a root-to-leaf path in a uniform binary tree of height N, where each branch independently has a cost of 1 or 0 with probability p and $1 - p$, respectively. We wish to prove that for $p > \frac{1}{2}$

$$P[M \geq (1 + \epsilon)C^*(N, p)] \to 0 \qquad \epsilon > 0,$$

where

$$M = \min_j \max_{n \in P_j^f} \{f(n)\},$$
$$f(n) = g(n) + \alpha^*[N - d(n)],$$

and where α^* is defined by the equation

$$\left(\frac{p}{\alpha^*}\right)^{\alpha^*} \left(\frac{1-p}{1-\alpha^*}\right)^{1-\alpha^*} = \frac{1}{2}.$$

Call a path P (α, L)-regular if the cost of every successive path segment of length L along P is at most αL. Karp and Pearl [9] have shown that

(a) $P[C^*(N, p) \leq \alpha N] \to 0$ for $\alpha < \alpha^*$.
(b) If $\alpha < \alpha^*$, one can always find a constant L_α such that the probability that there exists an (α, L_α)-regular path of length N is bounded away from zero. Call this probability $1 - q_\alpha$ $(q_\alpha < 1)$.

Consider the profile of f along a (α, L)-regular path P sprouting from level d_0 below the root. Along such a path $g(n)$ satisfies

$$g(n) \leq d_0 + [d(n) - d_0]\alpha + \alpha L,$$

and, consequently,

$$f(n) \leq (1 - \alpha)d_0 + \alpha^* N + \alpha L + d(n)(\alpha - \alpha^*).$$

For $\alpha > \alpha^*$, the expression on the right attains its maximum when $d(n)$ reaches its largest value of $N - d_0$, and so

$$M \leq \max_{n \in P} f(n) \leq (1 - 2\alpha + \alpha^*)d_0 + \alpha L + N\alpha.$$

Now let d_0 be any unbounded function of N such that $d_0 = o(N)$ and consider the probability $P[M \geq (1 + \delta)\alpha^* N]$. For every α between α^* and $(1 + \delta)\alpha^*$, the inequality

$$(1 - 2\alpha + \alpha^*)d_0(N) + \alpha L_\alpha + N\alpha \leq (1 + \delta)\alpha^* N$$

will be satisfied for sufficiently large N; hence, choosing $\alpha^* < \alpha < (1 + \delta)\alpha^*$, we have

$$P[M \geq (1 + \delta)\alpha^* N] \leq P[\text{no } (\alpha, L_\alpha)\text{-regular path stems from level } d_0(N)]$$
$$\leq (q_\alpha)^{2^{d_0(N)}} \to 0.$$

We can now bound our target expression by a sum of two probabilities:

$$P[M \geq (1+\epsilon)C^*(N,p)] \leq 1 - P\left[\left(\frac{1+\epsilon}{2}\right)C^*(N,p) \geq \alpha^* N \geq \left(\frac{1-\epsilon}{2}\right)M\right]$$

$$\leq P\left[C^*(N,p) \leq \frac{\alpha^* N}{1+\epsilon/2}\right] + P\left[M \leq \frac{\alpha^* N}{1-\epsilon/2}\right],$$

and, since each term on the right tends to zero, our claim is proved.

*Appendix B. Properties of A***

Algorithm A** is a variation of A* that uses an evaluation function

$$f'_{P_{s-n}}(n) = \max\{f(n') = g_{P_{s-n}}(n') + h(n') \mid n' \in P_{s-n}\}.$$

A** differs from A* in that it relies not only on the $g + h$ value of node n, but also considers the $g + h$ values along the path from s to n. The maximum is then used as a criterion for node selection. Note that A** cannot be considered a BF* algorithm since it uses one function f' for ordering nodes for expansion (step 3) and a *different* function g for redirecting pointers (step 6c). Had we allowed A** to use f' for both purposes, it would not be admissible relative to $\mathbf{I}_{AD}$, since f' is not order preserving.

We now show that A** is admissible over $\mathbf{I}_{AD}$.

THEOREM B1. *Algorithm A** will terminate with an optimal solution in every problem instance in which $h \leq h^*$.*

PROOF. Suppose the contrary. Let C be the value of the path P_{s-t} found by A** and assume $C > C^*$.

We argue that exactly before A** chooses the goal node t for expansion, there is an OPEN node n' on an optimal path with $f'(n') \leq C^*$. If we show that, than obviously A** should have selected n' for expansion and not t, since $f'(n') \leq C^* < C = f'(t)$, which yields a contradiction.

Since A** redirects pointers according to g, the pointer assigned to the shallowest OPEN node n' along any optimal path is directed along that optimal path. Moreover, $h \leq h^*$ implies that such a node satisfies $f'(n') \leq C^*$, and this completes our argument. □

We next show that A** dominates A* in the following sense.

THEOREM B2

(*a*) *For every tie-breaking rule of A* and for every problem instance $I \in \mathbf{I}_{AD}$, there exists a tie-breaking rule for A** that expands a subset of the nodes expanded by A*.*

(*b*) *Moreover, there exists a problem instance and a tie-breaking rule for A** that expands a proper subset of the nodes that are expanded by any tie-breaking rule of A*.*

PROOF

Part a. From the definition of f' it is clear that all paths that are strictly bounded below C^* relative to f are also strictly bounded below C^* relative to f'. Therefore, both algorithms have exactly the same set of surely expanded nodes, $\mathbf{N}_f^{C^*} = \mathbf{N}_{f'}^{C^*}$, and this set is expanded before any node outside this set. Let n^* be the first node expanded by A* satisfying the equality $f(n^*) = C^*$. Exactly before n^* is

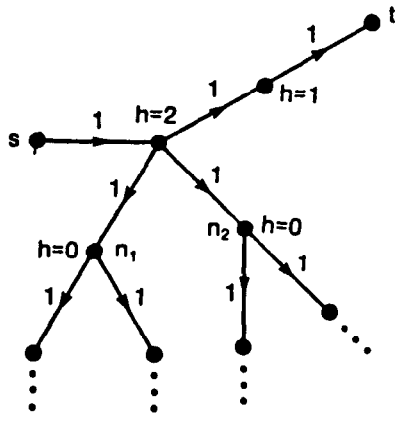


FIGURE 10

chosen for expansion, all nodes in $N_f^{C^*}$ are already expanded. A**, after expanding those nodes, also has n^* in OPEN with $f'(n^*) = C^*$; there exists, therefore, a tie-breaking rule in A** that will also choose n^* for expansion. From that moment on, A* will expand some sequence $n^*, n_1, n_2, n_3, \ldots, t$ for which $f(n_i) \leq C^*$. Since on these nodes $f'(n_i) = C^*$, it is possible for A** to use a tie-breaking rule that expands exactly that same sequence of nodes until termination.

Part b. Examine the graph of Figure 10. n_1 and n_2 will be expanded by every tie-breaking rule of A*, whereas there is a tie-breaking rule for A** that expands only P_{s-t}. □

REFERENCES

1. BAGCHI, A., AND MAHANTI, A. Search algorithms under different kinds of heuristics—A comparative study. *J. ACM 30,* 1 (Jan. 1983), 1–21.
2. BARR, A., AND FEIGENBAUM, E. A. *Handbook of Artificial Intelligence.* William Kaufman, Inc., Los Altos, Calif., 1981.
3. DREYFUS, S. E., AND LAW, A. M. *The Art and Theory of Dynamic Programming.* Academic Press, Orlando, 1977.
4. GELPERIN, D. On the optimality of A*. *Artif. Intell. 8,* 1 (1977), 69–76.
5. HARRIS, L. R. The heuristic search under conditions of error. *Artif. Intell. 5,* 3 (1974), 217–234.
6. HART, P. E., NILSSON, N. J., AND RAPHAEL, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern. SSC-4,* 2 (1968), 100–107.
7. HART, P. E., NILSSON, N. J., AND RAPHAEL, B. Correction to a formal basis for the heuristic determination of minimum cost paths. *SIGART Newsletter* (ACM) *37* (1972), 28–29.
8. IBARAKI, T. The power of dominance relations in branch-and-bound algorithms. *J. ACM 24,* 2 (Apr. 1977), 264–279.
9. KARP, R. M., AND PEARL, J. Searching for an optimal path in a tree with random costs. *Artif. Intell. 21,* 1 (Mar. 1983), 99–116.
10. LAWLER, E. L. AND WOOD, D. E. Branch-and-bound methods: A survey. *Oper. Res. 14,* 4 (1966), 699–719.
11. MARTELLI, A. On the complexity of admissible search algorithms. *Artif. Intell. 8,* 1 (1977), 1–13.
12. MÉRO, L. A heuristic search algorithm with modifiable estimate. *Artif. Intell. 23,* 1 (1984), 13–27.
13. NILSSON, N. J. *Problem-Solving Methods in Artificial Intelligence.* McGraw-Hill, New York, 1971.
14. NILSSON, N. J. *Principles of Artificial Intelligence.* Tioga, Palo Alto, Calif., 1980.
15. PEARL, J. On the discovery and generation of certain heuristics. *AI Magazine* (Winter/Spring 1983), 23–33.
16. PEARL, J. *HEURISTICS: Intelligent Search Strategies for Computer Problem Solving,* Addison-Wesley, Reading, Mass., 1984.
17. PEARL, J., AND KIM, J. H. Studies in semi-admissible heuristics. *IEEE Trans. Pattern Analysis Machine Intell. PAMI-4,* 4 (1982), 392–399.

18. POHL, I. First results on the effect of error in heuristic search. *Mach. Intell. 5* (1969), 219–236.
19. POHL, I. Bi-directional search. In *Machine Intelligence*, vol. 6, B. Meltzer and D. Michie, Eds. American Elsevier, New York, 1971, pp. 127–140.
20. POHL, I. The avoidance of (relative) catastrophe, heuristic competence, genuine dynamic weighting and computational issues in heuristic problem solving. In *Proceedings of the 3rd International Joint Conference on Artificial Intelligence* (Stanford, Calif., Aug. 20–23). William Kaufman, Inc., Los Altos, Calif., 1973.

RECEIVED MAY 1983; REVISED NOVEMBER 1984; ACCEPTED JANUARY 1985