Algorithm $R^*_{\delta,\epsilon}$

1. Select a risk measure and a risk tolerance δ.
2. Order nodes on OPEN by increasing values of the cost threshold $C_\delta(n)$, given by the solution to Eq. (3.23).
3. Form a sublist FOCAL of all nodes whose $C_\delta(n)$ deviates by at most ϵ from that of the leading node on OPEN, n_0:

$$\text{FOCAL} = \{n:\ C_\delta(n) - C_\delta(n_0) \leqslant \epsilon\}$$

4. Select (for expansion) a node from FOCAL that promises to facilitate the quickest search for the (optimal) completion part of the solution.
5. Halt when a node chosen for expansion is found to satisfy the goal conditions.

THEOREM 15. *Algorithm $R^*_{\delta,\epsilon}$ always finds a solution that does not exceed the cost threshold of all nodes in OPEN by more than ϵ.*

Proof: If $R^*_{\delta,\epsilon}$ terminates at node t, we have:

$$g(t) \leqslant C_\delta(t)$$

and since $t \in$FOCAL;

$$C_\delta(t) \leqslant C_\delta(n_0) + \epsilon \leqslant C_\delta(n) + \epsilon \qquad \forall\, n \text{ on OPEN}$$ ∎

Exceeding the cost threshold $C_\delta(n)$ by ϵ results in only a small increase in the overall termination risk. Additionally, we will show that for R_1 and R_3 the choice $\epsilon = \delta$ would lead to a termination risk of at most δ. Thus the speedup feature is obtained for free, without deteriorating the termination risk.

THEOREM 16. *For risk measures R_1 and R_3, algorithm $R^*_{\delta,\delta}$ is δ-risk-admissible.*

Proof: Assume first that $R^*_{\delta,\delta}$ terminates at node t. In Eq. (3.29) we obtained, for R_1 and R_3,

$$g(t) \leqslant C_\delta(t) - \delta.$$

t being in FOCAL implies

$$C_\delta(t) \leqslant C_\delta(n_0) + \epsilon \leqslant C_\delta(n) + \epsilon \qquad \forall\, n \text{ on OPEN}$$

and therefore

$$g(t) \leqslant C_\delta(n) + \epsilon - \delta \qquad \forall\, n \text{ on OPEN}.$$

The choice $\epsilon = \delta$ leads to

$$g(t) \leqslant C_\delta(n) \qquad \forall\, n \text{ on OPEN}$$

which is the condition for δ-risk-admissibility. ∎

Pearl, Judea. Heuristics: Intelligent Search Strategies for Computer Problem Solving. Addison-Wesley, 1984.

Proof of R_δ* Termination with Risk Measures R_1, R_2, or R_3. The proof consists of two steps. The first step shows that if a solution path exists, then at all times OPEN contains a node on this path for which $R_\zeta(n)$ is bounded. The second step shows that any such node cannot remain unexpanded for all but a finite number of steps. The two steps of the proof are:

1. From the definition of $R_1(C) = C - g - h_a$, it is clear that the equation $R_1(C) = \delta$ has a finite solution $C_\delta = \delta + g + h_a$, and therefore, $C_\delta(n)$ is finite for all nodes on OPEN. Likewise, the equation $R_2(C) = \delta$ has a solution $C_\delta \leqslant 1$ for all $0 \leqslant \delta \leqslant 1$. The finiteness of C_δ for the expected risk $R_3(C)$ is based on the assumption that the density $\rho_h(x)$ possesses a finite expectation $E(h) < \infty$ for every node on a solution path. With this assumption, we can write

$$R_3(C) \geqslant C[1 - P(f^+ > C)] - E(f^+)$$

and, using Tchebycheff's inequality

$$R_3(C) \geqslant C - 2E(f^+) = C - 2g - 2E(h)$$

Clearly the equation $R_3(C) = \delta$ also possesses a finite solution C_δ.
2. The inequality $C_\delta(n) \geqslant g(n)$ holds for each node on OPEN since the decision to abandon n after finding a solution with cost $C < g(n)$ carries no risk at all. Now, since $g(\cdot)$ was assumed to increase beyond bounds along any infinite path (see proof of Theorem 1), R_δ* must return after a finite number of expansions to those nodes on OPEN that, by virtue of their belonging to some solution path, possess bounded C_δ.

3.3 SOME EXTENSIONS TO NONADDITIVE EVALUATION FUNCTIONS (*BF** AND *GBF**)

Our discussions in this chapter have so far been limited to the additive cost measure $g(n') = g(n) + c(n, n')$ and to additive evaluation functions such as $f = (1-w)g + wh$. In this section, our aim is to examine how the results established in Section 3.1 will change if we remove both restrictions. The minimization objective 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 lowest branch cost along the path), the cost of the **last** node, and many others. Additionally, even in the usual case where the minimization objective is the 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. 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 another example, the additive evaluation $f = g + h$ with h an arbitrary nonadmissible estimate of h^*.

3.3.1 Notation and Preliminaries

The computations required by some of these generalizations may no longer be executable by A^* and we may have to employ more elaborate best-first algorithms (see Figure 2.10) such as BF^* (in case of nonrecursive cost measures) or even GBF^* (in case of cost measures that are not order-preserving as in Figure 3.5). In the following discussion, we use the terminology of the BF^* algorithm (i.e., delayed termination with irrevocable parent selection) and hope that the reader can infer the modifications needed if GBF^* is employed. For comparison purposes, we assign to each new result an asterisk number (e.g., Theorem 1*, Theorem 2*) pointing to the corresponding topics in Section 3.1.

The objective of BF^* is to find an optimal or a near-optimal solution path in a directed locally finite graph, where each solution path $P^s = s, n_1, n_2, \ldots, \gamma, \gamma \in \Gamma$, is assigned a cost measure $C(P^s)$ which may be a complex function of all the nodes and the arcs along P^s. In principle, the cost measure $C(P^s)$ need only be defined on complete solution paths in $\mathbf{P}_{s,\Gamma}$, as only these paths are returned by the search algorithm at termination. However, since every node may be proclaimed a goal node in some conceivable problem instance, the domain of $C(\cdot)$ essentially spans all path-segments in G which begin at s.

To each node n_i along a given path $P = s, n_1, n_2, \ldots, n_i, \ldots$ we assign a nonnegative evaluation function $f_P(n_i)$ that *depends only on the portion of the path leading to that node.* Thus, $f_P(n_i)$ is a shorthand notation for $f(s, n_1, n_2, \ldots, n_i)$. As the search progresses, a given node n may be assigned to different paths depending on the pointers used by BF^* to trace back the path from s to n. However, at any given time the computation of $f_P(n)$ is uniquely determined by the pointer-path assigned to n at that time, and so $f(n)$ will denote an arbitrary element in the set $\{f_P(n): P \in \mathbf{PP}\}$ where $\mathbf{PP}$ is the set of all pointer-paths constructed during the execution of the search. Moreover, BF^* adjusts pointers along the path of lowest f. Hence, if P_1 and P_2 are two paths leading to n from two distinct parents of n, we can write $f_{P_1}(n) \leqslant f_{P_2}(n)$ if path P_1 was explicated after P_2. In particular, $f_{P_i}(n) \leqslant f(n)$ if P_i is known to be the path assigned to n at termination.

This is a convenient point to demonstrate why an adequate analysis of BF^* on graphs that are not trees requires that f be **order-preserving.** Relations such as $f_{P_i}(n) \leqslant f(n)$ would be very helpful if $f(n)$ stood, not merely for the set of values attached to n during the execution of the search algorithm, but also for

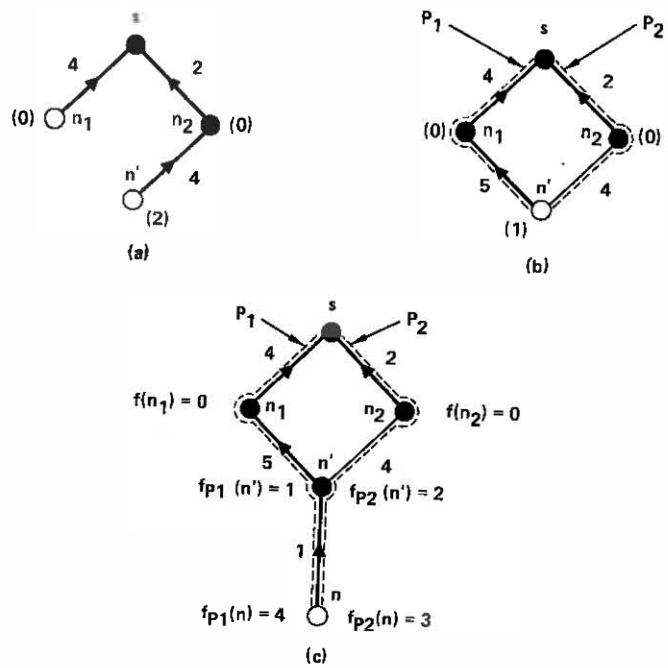

Figure 3.5
An order-reversal produced by using the range as the cost measure, i.e., $f_P(n)$ stands for the difference between the highest and lowest edge values along the path P leading to n.

the entire set of values n *may receive* along each one of the paths leading to n in the graph G_e explicated by the algorithm. Figure 3.5 exemplifies the difference between the two sets using the cost-range measure as an f function that is not order-preserving.

In Figure 3.5(*b*) a new path, P_1 is discovered to n' and, since $f_{P_1}(n') = 1 < f_{P_2}(n') = 2$, BF^* redirects the pointer of n' along P_1. Next, when n is generated, BF^* assigns to it a pointer back to n', that is, along $P_1 = s, n_1, n', n$, while $P_2 = s, n_2, n', n$ is no longer considered. However, the value of $f_{P_1}(n)$ at this point is 4, whereas the f value along P_2 would have been lower since $f_{P_2}(n) = 3$. Thus, even though the two paths to n are part of the subgraph G_e explicated by BF^* with $f_{P_1}(n) > f_{P_2}(n)$, we find that n is currently directed along the inferior path P_1. Such an order reversal in the ranking of P_1 and P_2 would not have occurred if f was an order-preserving function such as the additive, multiplicative, or maximum-cost measures.

Likewise, the relation $f_{P_1}(n) > f_{P_2}(n)$ would be recognized by *GBF** which would have maintained both $P_1 = s, n_1, n', n$ and $P_2 = s, n_1, n', n$ in the list of candidate solution bases and, as soon as n was generated, would have computed $f_{P_1}(n) = 4$ and and $f_{P_2}(n) = 3$ and would have reassigned the pointer of n' back along P_2.

DEFINITION: *Let $P_i P_j$ denote the concatenation of two paths in a graph, i.e., P_j is an extension of P_i. An evaluation function f is said to be* ***order-preserving*** *if for any two paths P_1 and P_2 leading from s to n, and for any extension P_3 of those paths, the following holds:*

$$f(P_1) \geqslant f(P_2) \Rightarrow f(P_1P_3) \geqslant f(P_2P_3)$$

In terms of our $f_P(n)$ notation, order-preservation can be written

$$f_{P_1}(n) \geqslant f_{P_2}(n) \Rightarrow f_{P_1P_3}(n') \geqslant f_{P_2P_3}(n') \qquad n' \in P_3$$

Order-preservation is a version of the **principle of optimality** in dynamic programming (Dreyfus and Law, 1977), and it simply states that if a path P_1 from s to n is judged to be more meritorious than another path P_2, also from s to n, then no common extension of P_1 and P_2 may later reverse this judgment. In other words, the information that may be gathered by exploring node n has no bearing on the relative merit of candidate path-segments from s to n and hence *BF**'s decision to irrevocably discard the less meritorious parent as soon as n is generated in duplicate is justified. The discarded parent cannot possibly offer a better path to any of n's descendants. Based on this property, we may state the following lemma.

LEMMA 0*: *If f is order-preserving and P_1 and P_2 are any two paths such that n is currently directed along P_1 and P_2 has been explicated in the past (i.e., all arcs along P_2 have been generated), then*

$$f_{P_1}(n) \leqslant f_{P_2}(n)$$

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

$$f_{P_i}(n) \leqslant f_P(n) \qquad \forall\, P \in G_n \qquad \blacksquare$$

Note that for *GBF** the lemma holds true even without the requirement that f be order-preserving.

Relations between C and f. 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 bases. 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,

$$f(s, n_1, n_2, \ldots, \gamma) = \psi[C(s, n_1, n_2, \ldots, \gamma)] \qquad \gamma \in \Gamma \qquad (3.31)$$

where ψ is an increasing function of its argument. 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 $h(\gamma) = 0$ guaranteed the identity $f = C$ on solution paths, whereas on other paths f was not governed by C since, in general, h could take on arbitrary values.

3.3.2 Algorithmic Properties of Best-First Search *BF**

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 shall 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,

$$M_j = \max_{n \in P_j^s} \{f_j(n)\} \qquad (3.32)$$

and let M be the minimum of M_j:

$$M = \min_j \{M_j\} \qquad (3.33)$$

Henceforth we will assume that both the max and the min functions are well defined.

Termination* and *Completeness

LEMMA 1*: *At any time before BF* terminates, there exists on OPEN a node n' that is on some solution path and for which $f(n') \leqslant M$.*

Proof: Let $M = M_j$, i.e., the min-max 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:

$$M_j = \max_{n_i \in P_j^s} \{f_j(n_i)\}$$

therefore,

$$f_j(n') \leqslant M_j = M$$

Moreover, since all ancestors of n' on P_j^s are on CLOSED and BF^* has decided to assign its pointers along P_i^s, Lemma 0* states

$$f_i(n') \leqslant f_j(n')$$

This implies

$$f_i(n') \leqslant M$$

which proves Lemma 1*. ∎

LEMMA 2*: *Let P be any pointer path established by BF^* at termination time. Then any time before termination there is an OPEN node n on P for which $f(n) = f_P(n)$.*

Proof: Let n be the shallowest OPEN node on P at some arbitrary time t before termination. Since all n's ancestors on P are closed at time t, P was explicated and, hence (Lemma 0*), n must be assigned an f at least as cheap as $f_P(n)$. Thus $f_P(n) \geqslant f(n)$ with strict inequality holding only if, at time t, n is found directed along a path different than P. However, since BF^* eventually terminates with pointers along P, it must be that BF^* has never encountered another path to n with cost lower than $f_P(n)$. Thus $f(n) = f_P(n)$. ∎

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

Proof: The proof is similar to that of Theorem 1 in Section 3.1.2, using the boundness of $f(n') \leqslant M$, $n' \in OPEN$. ∎

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^*.

Properties of the Final Solution Path

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 1* we learn that BF^* cannot select for expansion any node n having $f(n) > M$. This includes the node $t \in \Gamma$ and, hence, $f_j(t) \leqslant M$. But Eq. (3.31) implies that $f_j(t) = \psi[C(P_j^s)]$ and so, since ψ and ψ^{-1} are monotonic,

$$C(P_j^s) \leqslant \psi^{-1}(M)$$

which proves the theorem. ∎

Theorem 2* can be useful in appraising the degree of suboptimality exhibited by nonadmissible algorithms. For example, Pohl's dynamic weighting scheme (Section 3.2.2, Eq. (3.19)) can be easily shown to be ϵ-admissible. Indeed, for the evaluation function

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

$h(\gamma) = 0$ and Eq. (3.31) dictate $\psi(C) = C$ and we can bound M by considering $\max_n f_{P^*}(n)$ along any optimal path P^*. Thus

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

On the other hand, Theorem 2* states that the search terminates with cost $C_t \leqslant M$. Hence

$$C_t \leqslant C^*(1 + \epsilon)$$

This example demonstrates that any evaluation function of the form

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

will also be ϵ-admissible, as long as $\rho_P(n) \leqslant 1$ along some optimal path P^*. In more elaborate cases where it is impossible to guarantee an upperbound to M, the statistical properties of M may be used to assess the average degree of suboptimality, $E(C_t - C^*)$; see exercise 5.4.

Theorem 2* can also be used to check for ordinary admissibility, i.e., $C_t = C^*$; 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 upperbound on f along *any* solution path constitutes an upperbound on M.
2. The relation between $f_P(n)$ and C^* is more transparent along an *optimal path*. (For example, in the case of A^* with $h \leqslant h^*$, the relation $f_{P^*}(n) \leqslant C^*$ 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 $P_j^s = s, ..., t$ 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 $f_i(t) \leqslant M$. Moreover, from the definition of M we have

$$M \leqslant \max_{n \in P_i^s} f_i(n)$$

for every solution path P_i^s. In particular, taking $P_i^s = P^*$, we obtain

$$f_i(t) \leqslant M \leqslant \max_{n \in P} f_P(n) = f_{P^*}(\gamma)$$

However, from Eq. (3.31) we know that f is monotonic in C when evaluated on complete solution paths, thus

$$C(P_j^s) \leqslant C^*$$

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

The admissibility condition for Pohl's weighted heuristic $f_w = (1 - w)g + wh$ (Section 3.2.1) can be obtained directly from Corollary 1. Here $\psi(C) = (1 - w)C$ which complies with Eq. (3.31) 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

$$f_{P^*}(n) \leqslant f_P(\gamma)$$

we obtain

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

or

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

Clearly, if the ratio $h(n)/h^*(n)$ is known to be bounded from above by a constant β, then setting $1 - w/w \geqslant \beta$ will satisfy this inequality. Therefore, the condition

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

delineates the range of admissibility of A^* and, for $h(n) \leqslant h^*(n)$ ($\beta = 1$), it simply becomes $w \leqslant \frac{1}{2}$. Note, however, that the use of $w > \frac{1}{2}$ may also be admissible if h is known to consistently underestimate h^* such that $h(n)/h^*(n) \leqslant \beta < 1$. Conversely, if h is known to be nonadmissible with $\beta > 1$, the use of $w = 1/(1+\beta)$ will turn A^* admissible. This technique is called **debiasing** and its effect on the average run-time of A^* is analyzed in Chapter 7.

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 the usual task of minimizing an additive cost measure. If $h \leqslant h^*$ and f is monotonic in both arguments, then Corollary 1 states that $f(g, h)$ is guaranteed to be admissible as long as

$$f(g, C - g) \leqslant f(C, 0) \qquad \text{for } 0 \leqslant g \leqslant C$$

Thus, for example, $f = \sqrt{g^2 + h^2}$ is admissible while $f = (g^{1/2} + h^{1/2})^2$ 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.

It can be shown, however, that of all functions $f(g, h)$, $f = g + h$ results in the least number of nodes expanded (Dechter and Pearl, 1983) and, in this sense, A^* can be termed **optimal**. If, in addition, h is consistent, then A^* largely dominates *every* admissible algorithm (see exercise 3.7), even those outside the best-first class.

Conditions for Node Expansion

LEMMA 3*: *Let P_j^s be the solution path with which BF^* terminates, then M is obtained on P_j^s, i.e., $M = M_j$.*

Proof: Suppose BF^* terminates on P_j^s, but $M_j > M$, and let $n^* \in P_j^s$ be such that $f_j(n^*) = M_j$. 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 $f(n') \leqslant M$ (Lemma 1*), and so

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

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

COROLLARY 2: *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 $n^* \in P_j^s$ 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,

$$f(n^*) = f_j(n^*) = M_j = M$$ ∎

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

Proof: Follows directly from Lemma 1*. ∎

THEOREM 4*. *Let n^* be the first node with $f(n^*) = M$ which is expanded*

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

Proof: $f(n)$ can only decrease through the redirection of 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^*. ∎

Note the difference between Theorems 3* and 4* and Theorems 3 and 4 in Section 3.1. First, M plays the role of C^* on the right-hand side of the inequalities. Second, the sufficient condition for expansion in Theorem 4* must also include the provision that n enters OPEN before the expansion of n^*. This provision was not necessary in Theorem 4 because C^* is always obtained on the terminal node t and so, if a node n ever resides on OPEN, it automatically enters OPEN before t's expansion. When f is nonadditive (or even additive with nonadmissible h), it is quite possible that M will be obtained on a nonterminal node n^* and then, a descendant n of n^* may enter OPEN satisfying $f(n) < f(n^*) = M$ and still will not be expanded.

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

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 (M cannot be obtained on s). 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*, n will be expanded before n^*. ∎

The final results we wish to establish are necessary and sufficient conditions for node expansion which 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 path $P_{n_i - n}$ where*

$$L_i = \max_{k \geq i} f_j(n_k).$$

In other words, there should exist a path $P_{n_i - n}$ along which

$$f(n') \leqslant \max_{k \geq i} f_j(n_k) \qquad \forall n' \in P_{n_i - n} \tag{3.34}$$

Moreover, a sufficient condition for expanding n is that (3.34) 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 3.6). Since P_j^s is the solution path eventually found by BF^* we know (see proof of Lemma 2*) that at time t_n n_k is *along* P_j^s and, therefore,

$$f(n) \leqslant f(n_k) = f_j(n_k)$$

We are now ready to identify the node n_i on P_j^s that satisfies (3.34). 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, $f(n) \leqslant f_j(n_k)$ implies

$$f(n) \leqslant \max_{k \geq i} f_j(n_k)$$

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'

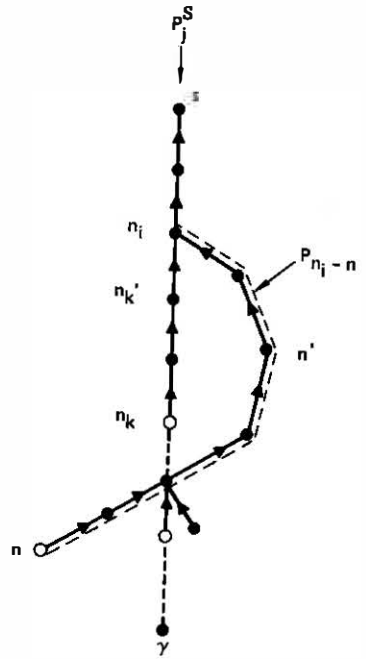

Figure 3.6
The condition for expanding n given that path P_j^s is eventually found by the search. n will be expanded if all the f values along path $P_{n_i - n}$ are lower than the maximal f value between n_i and γ (along P_j^s).

may have encountered a different $n_{k'}$ on OPEN, but each such $n_{k'}$ must have been a descendant of n_i along P_j^s satisfying $f(n') \leqslant f_s(n_{k'})$. Hence, Eq. (3.34) 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 proven by assuming that $\max_{k \geq i} f_j(n_k)$ 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

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

However, since Eq. (3.34) is assumed to hold with strict inequality for any n' along $P_{n_i - n}$, we must conclude

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

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 Chapter 7 where we analyze the average complexity of nonadmissible algorithms, treating f as a random variable.

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 Eq. (3.34) 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 search 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 by Dechter and Pearl (1983), but it is of a rather complex form when f is nonmonotonic. Fortunately the model we analyze in Chapter 7 is a tree having only one solution path and so, since the identity of P_j^s is given, the expansion condition of Theorem 6* becomes extremely simple (see Figure 7.1).

3.4 BIBLIOGRAPHICAL AND HISTORICAL REMARKS

A formal description of A^* and many of its properties were presented in a paper by Hart, Nilsson, and Raphael (1968). The fact that the performance of A^* cannot improve by making h lower (Theorem 7) was originally thought to depend on the consistency property of h. This error was corrected in Hart, Nilsson and Raphael (1972) and in Nilsson (1980). Our treatment, using the concept of ϵ-bounded paths, renders the proof of Theorem 7 simpler than in earlier works. The property of monotonicity was introduced by Pohl (1977) to replace that of consistency. Surprisingly, the equivalence of the two (Theorem 8), as well as Theorem 9 (that consistency implies admissibility), have not previously been noted in the literature.

The **optimality** of A^*, in the sense of expanding the least number of distinct nodes, has been a subject of some confusion. Theorem 7 has often been interpreted to reflect some supremacy of A^* over other search algorithms of equal information, and, consequently, several authors have assumed that A^*'s optimality is an established fact (e.g., Nilsson 1971, Méro 1981, Barr and Feigenbaum, 1981). In fact, all Theorem 7 says is that some A^* algorithms are better than other A^* algorithms. 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 using the entire information gathered along each path. These two conjectures have been examined by Gelperin (1977) and Dechter and Pearl (1983) and were finally given a qualified confirmation using the results of Section 3.3.

Gelperin (1977) has correctly pointed out that in any discussion of the optimality of A^* one should compare it to a wider class of equally informed algorithms, not merely those guided by $f = g + h$, and that the comparison class should include, for example, algorithms which *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 over-restrictive definition of *equally informed*. Gelperin's interpretation of "an algorithm B is *never more informed* than A" not only restricts B from using information unavailable to A, but also 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 restricted from ever assigning to a node n an $f_B(n)$ value higher than A would, even if the information gathered along the path to n justifies such an assignment.

Dechter and Pearl (1983) used the more natural interpretation 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. They considered the class of all search algorithms seeking a lowest (additive) cost path in a graph where the nodes are assigned an arbitrary heuristic function h, and assumed that $h(n)$ is made available to any algorithm that generates node n. Within this class, they first showed that A^* is optimal over all those algorithms which, for all h's, are guaranteed to find a solution at least as good as A^*. Second, they considered the wider class of algorithms which like A^* are only admissible whenever $h \leqslant h^*$ and showed that on this class A^* is not optimal and that no optimal algorithm exists unless h is also restricted to be consistent (exercise 3.7a and 3.7b). Finally, they showed that A^* is optimal over the subclass of *best-first* algorithms (i.e., *BF**) which are admissible when $h < h^*$.

Section 3.2 is based on Pearl and Kim (1982); other methods of relaxing the admissibility requirement to gain speed are reported by Harris (1974), Pohl (1973), and Ghallab (1982). Section 3.3 is based on Dechter and Pearl (1983), although some of the results were also reported by Bagchi and Mahanti (1983). Improvements in A^* which reduce the number of node reopenings in the case of nonmonotone h were introduced by Martelli (1977), Méro (1981), and Bagchi, et al. (1983); see exercise 3.5.