Multiobjective A^*

BRADLEY S. STEWART AND CHELSEA C. WHITE, III

University of Virginia, Charlottesville, Virginia

Abstract. A multiobjective generalization of the heuristic search algorithm A^* is presented, called MOA^*. The research is motivated by the observation that most real-world problems involve multiple, conflicting, and noncommensurate objectives. MOA^* explicitly accommodates this observation by identifying the set of all nondominated paths from a specified start node to a given set of goal nodes in an OR graph. It is shown that MOA^* is complete and, when used with a suitably defined set of admissible heuristic functions, admissible. Several results concerning the comparison of versions of MOA^* directed by different sets of heuristic functions are provided. The implications of using a monotone or consistent set of heuristic functions in MOA^* are also discussed. Two simple examples are used to illustrate the behavior of the algorithm.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*computations on discrete structures*; *routing and layout*; *sorting and searching*; G.2.1 [**Discrete Mathematics**]: Combinatorics—*combinatorial algorithms*; G.2.2 [**Discrete Mathematics**]: Graph Theory—*graph algorithms*; *network problems*; *path and circuit problems*; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*dynamic programming*; *graph and tree search strategies*; *heuristic methods*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: A^* multiobjective decisionmaking

1. *Introduction*

Many, if not most, problem-solving approaches can be interpreted as procedures for iteratively searching through a set of predetermined or progressively determined solution alternatives until a satisfactory, perhaps in some sense optimal, solution is found. The Artificial Intelligence (AI) community has placed special emphasis on this view of problem solving as search. AI search procedures are typically formulated in terms of a graph containing nodes representing potential problem states. The objective is to efficiently determine a "best" path from a designated start node to a member of a distinguished set of goal nodes. The criteria used to identify a "best" path have invariably been scalar-valued.

The research on this paper was partially supported by the Army Research Office under contract DAAG-29-85 K0089 and the Institute for Parallel Computation at the University of Virginia.

Authors' addresses: B. S. Stewart, Bellcore, MRE-IF210, 435 South Street, Morristown, NJ 07960; C. C. White, III, Department of Systems Engineering, School of Engineering and Applied Science, Thornton Hall, University of Virginia, Charlottesville, VA 22901.

Most reasonably complete formulations of real-world problems involve multiple, conflicting, and noncommensurate objectives. As an example, Donald [8] recently identified a multiobjective search problem in the area of robot planning. The task of adequately modeling such problems using a single scalar-valued criterion has been the subject of considerable research, as discussed by Keeney and Raiffa [13] and other researchers in the area of multiattribute utility theory (MAUT). Difficulties associated with accurately and confidently determining such a scalar-valued criterion on which to base the selection of a most preferred alternative, discussed, for example, by Zeleny [33], have led to the development of the multiobjective approach to alternative selection.

The multiobjective approach models each objective by a scalar-valued criterion. Then, in contrast to the MAUT approach in which a single scalar-valued function of these criteria is determined and an alternative that optimizes this function is sought, the less ambitious goal becomes to determine the set of nondominated alternatives. An alternative α is said to be *nondominated* among a set of alternatives if there is no other alternative in the set that is (1) at least as "good" as α with respect to all of the objectives, and (2) strictly "better" than α with respect to at least one of the objectives, where, "good" and "better" are defined in terms of the scalar-valued criteria associated with individual objectives. (See also definitions below.) Thus, the multiobjective approach allows the alternative selection criterion to be vector-valued and identifies all clearly inferior alternatives so that they may be excluded from further consideration. The design, application, and evaluation of procedures for determining the "most preferred" alternative from a set of nondominated alternatives is currently a topic of considerable research interest; see, for examples, the following: [4], [5], [14], [15], [16], [17], [19], [21], [29], and [31].

Our initial efforts have focused on producing a multiobjective generalization of A^*, an important AI search procedure originally presented by Hart et al. [10], with corrections provided in [11]. Other important references on A^* include the works by Nilsson [22, 23] and Pearl [24]. We refer to our algorithm as multiobjective A^*, or MOA^*. The initial focus on A^* is due to its simple, yet powerful, structure, which facilitates analysis of its performance. In this paper, we outline the MOA^* algorithm, illustrate its application to a simple multiobjective shortest path problem, and examine some of its formal properties.

2. *Overview of MOA**

In this section, we present an outline of the MOA^* algorithm and a simple example of its use. We refer to MOA^* as a heuristic search algorithm due to the fact that it is guided by a set of functions whose values may not precisely indicate any properties of the solution space. The algorithm is also unreliable in that, in cases where the guiding heuristics do not meet a certain bounding test (admissibility—see Section 3.3 below), it is not guaranteed to produce a useful solution to the multiobjective search problem. However, when used with the proper set of heuristics, MOA^* is completely reliable, in a sense described in Section 4.3 below.

2.1. ALGORITHM OUTLINE. The description of MOA^* in Figure 1 is intended to provide an intuitive grasp of the algorithm and its top-level structure.

```
0. Initialize by setting OPEN equal to a set containing only the start node and setting CLOSED,
   SOLUTION, SOLUTION_COSTS, SOLUTION_GOALS, and LABEL each equal to the
   empty set.
1. Find the set of nodes in OPEN, call it ND, that have at least one node selection function value that
   is not dominated by:
   1.1. the cost of any solution path already discovered (i.e., in SOLUTION_COSTS), nor by
   1.2. the node selection function values of any other potential solution represented by a node on
        OPEN.
2. Terminate or select a node for expansion.
   2.1. If ND is empty, do the following:
        2.1.1. Use the set of preferred solution path costs in SOLUTION_COSTS and the
               LABEL sets, if any, to trace through backpointers from the goal nodes in
               SOLUTION_GOALS to $s$.
        2.1.2. Place any solution paths in SOLUTION.
        2.1.3. Stop.
   2.2. Otherwise, do the following:
        2.2.1. Use a domain-specific heuristic to choose a node $n$ from ND for expansion, taking
               goals, if any, first.
        2.2.2. Remove $n$ from OPEN.
        2.2.3. Place $n$ on CLOSED.
3. Do bookkeeping to maintain accrued costs and node selection function values.
4. Identify solutions.
   4.1 If $n$ is a goal node, do the following:
        4.1.1. Add it to SOLUTION_GOALS.
        4.1.2. Add its current costs to SOLUTION_COSTS.
        4.1.3. Remove any dominated members of SOLUTION_COSTS.
        4.1.4. Go to Step (6).
   4.2. Otherwise, continue.
5. Expand $n$ and examine its successors.
   5.1 Generate the successors of $n$.
   5.2 If $n$ has no successors, go to Step 6.
   5.3 Otherwise, for all successors $n'$ of $n$, do the following:
        5.3.1 If $n'$ is a newly generated node, do the following:
           5.3.1.1. Establish a backpointer from $n'$ to $n$.
              5.3.1.1.1. Set $LABEL(n', n)$ equal to the nondominated subset of the set of accrued
                         costs of paths through $n$ to $n'$ that have been discovered so far.
           5.3.1.2. Establish a nondominated accrued cost set, $G(n') = LABEL(n', n)$.
           5.3.1.3. Compute node selection values, $F(n')$, using $G(n')$ and the heuristic function
                    values at $n'$, $H(n')$.
           5.3.1.4. Add $n'$ to OPEN.
        5.3.2. Otherwise, $n'$ was previously generated, so do the following:
           5.3.2.1. If any potentially nondominated paths to $n'$ have been discovered, then, for each
                    one, do the following:
              5.3.2.1.1. Ensure that its cost is in $LABEL(n', n)$ and therefore in the current set of
                         nondominated accrued costs of paths discovered so far to $n'$; that is, in $G(n')$.
              5.3.2.1.2. If a new cost was added to $G(n')$, do the following:
                 5.3.2.1.2.1. Purge from $LABEL(n', n)$ those costs associated with paths to $n'$ to
                              which the new path is strictly preferred.
                 5.3.2.1.2.2. If $n'$ was on CLOSED, move it to OPEN.
6. Iterate.
   6.1. Increment iteration counter.
   6.2. Go to Step (1).
```

FIG. 1. Outline of the *MOA** algorithm.

The use of sets *OPEN* and *CLOSED* for control of the search is typical in heuristic search work. The necessary generalization of the various functions related to node evaluation to allow for a set of evaluations at any given node results in the use of the sets G, H, and F. The requirement for accommodating

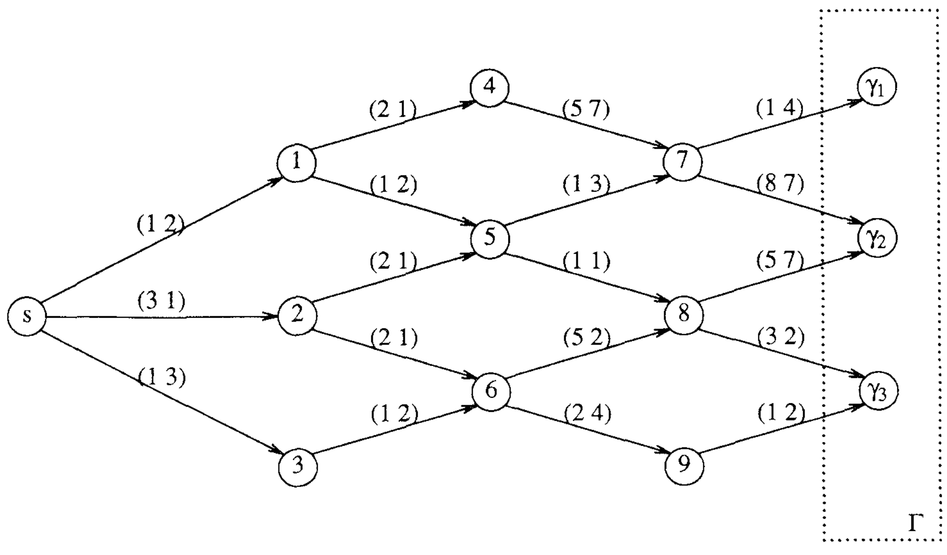


FIG 2. State-space graph for Example 1

possible multiple backpointers at a node results in the use of the *LABEL* sets. *SOLUTION* and *SOLUTION_GOALS* are introduced to help collect multiple solutions, and *SOLUTION_COSTS* is necessary in our general case to account for the fact that there may be a number of nondominated paths with distinct costs. Finally, *ND* is introduced as a subset of *OPEN* to be identified for particular attention in the process of selecting a node for expansion. Precise definitions of these sets can be found in Section 3.3 below.

Note that if MOA^* is applied to a single objective problem, then its operation is the same as that of a version of A^* modified to find all optimal paths. Note further that if Step (4) of MOA^* is altered by changing the phrase "go to Step (6)" to "stop," then, when used on single objective problems, MOA^* will operate the same as A^*. Several corollaries in Section 4 are direct consequences of this correspondence.

2.2. EXAMPLE 1. The following multiobjective shortest path problem illustrates the operation of the MOA^* algorithm on a very simple state-space graph. The purpose of this example is to present an overview of the general behavior of the algorithm without belaboring the computational aspects. More detail on the algorithm is given below. The computational complexity of MOA^* in particular, and of multiobjective search algorithms in general, represent interesting topics for further research.

A state-space graph is shown in Figure 2, which also shows the values of the arc costs for Example 1. The value of M (number of objectives) for this simple example is 2. Based on the information in the figure, we may calculate the derived cost information that is shown in Table I (see Section 3.3 for definitions of terms and quantities). Also shown in Table I are the heuristic values used for this illustration. The heuristic values were simply defined by the nondominated members of the set of costs corresponding to the arcs with tails at a node. For example, the set of costs of arcs with tails at the start node is $\{(1\ 2), (3\ 1),$

TABLE I. Derived Costs and Next-Arc-Cost Heuristic Function Values for Example 1

	Derived Costs			Heuristic Values
Node	$G^*(n)$	$H^*(n)$	$F^*(n)$	$H(n)$
s	(0 0)	(4 11) (6 7) (9 5)	$H^*(s)$	(1 2) (3 1)
1	(1 2)	(3 9) (5 5)	(4 11) (6 7)	(2 1) (1 2)
2	(3 1)	(4 8) (6 4) (5 7)	(7 9) (9 5) (8 8)	(2 1)
3	(1 3)	(9 6) (4 8)	(10 9) (5 11)	(1 2)
4	(3 3)	(6 11)	(9 14)	(5 7)
5	(2 4) (5 2)	(2 7) (4 3)	(4 11) (6 7) (9 5) (7 9)	(1 1)
6	(5 2) (2 5)	(8 4) (3 6)	(13 6) (8 8) (5 11) (10 9)	(5 2) (2 4)
7	(3 7) (6 5)	(1 4)	(4 11) (7 9)	(1 4)
8	(3 5) (6 3)	(3 2)	(6 7) (9 5)	(3 2)
9	(7 6) (4 9)	(1 2)	(8 8) (5 11)	(1 2)
γ_1	(4 11) (7 9)	(0 0)	(4 11) (7 9)	(0 0)
γ_2	(8 12) (11 10)	(0 0)	(8 12) (11 10)	(0 0)
γ_3	(6 7) (9 5)	(0 0)	(6 7) (9 5)	(0 0)

(1 3)}, of which one is dominated so the heuristic values for the start node are {(1 2), (3 1)} as shown in Table I. These sets of heuristic function values are clearly admissible (see definition in Section 4.3) since they have the necessary lower bounding property with respect to actual costs of partial solution paths.

Operation of the algorithm with the next-arc-cost heuristic is easily followed using the sequence of 11 graphs that make up Figure 3 and the corresponding trace in Table II. Each graph illustrates the state of the search for an iteration of the algorithm. Arcs of the initial problem state-space OR graph are shown as dashed lines. As the arcs are traversed during the search, they are made solid in

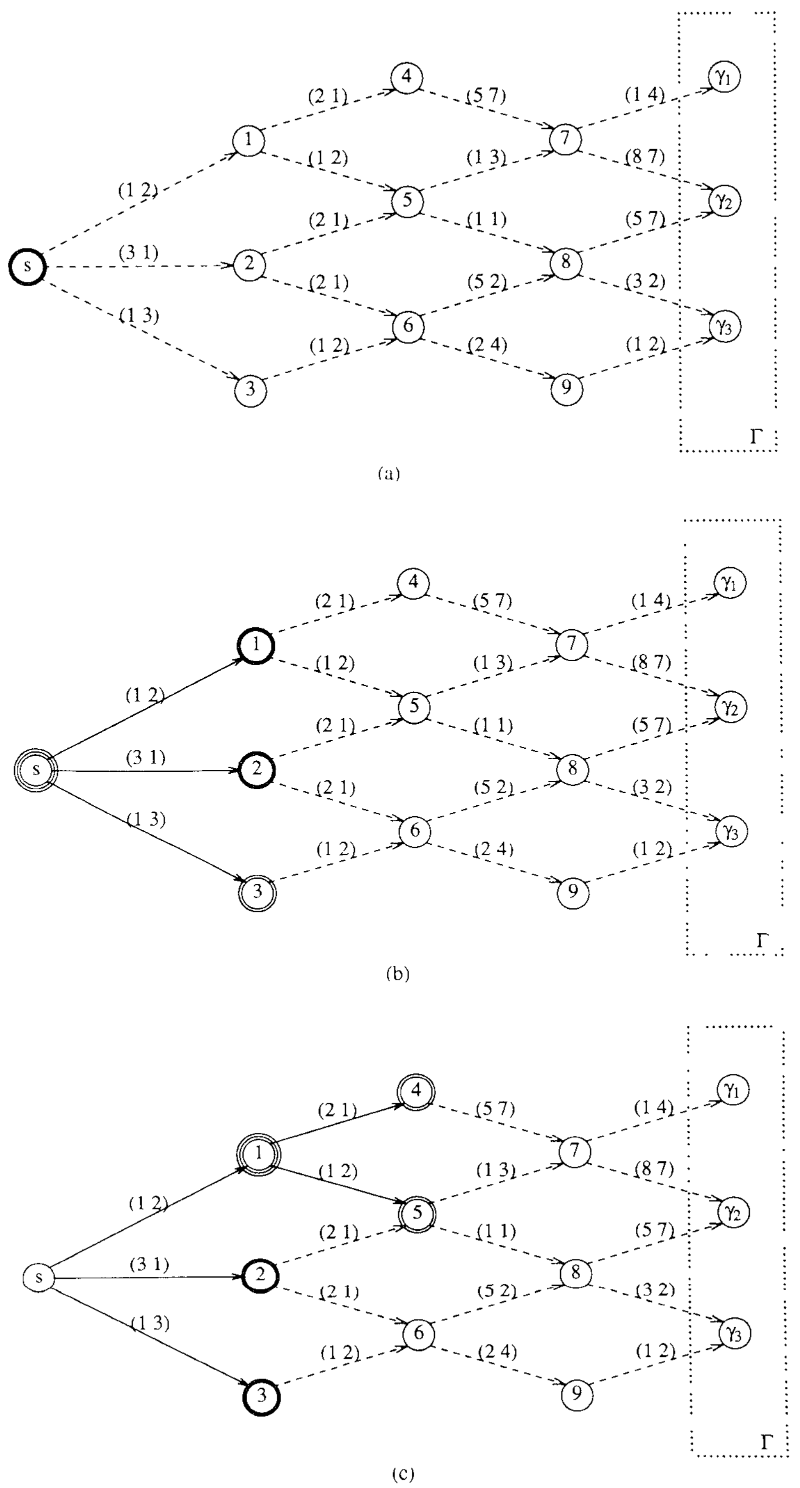
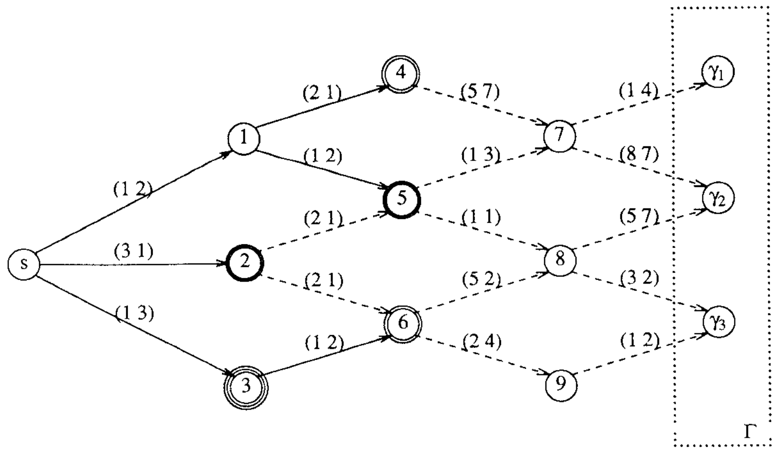
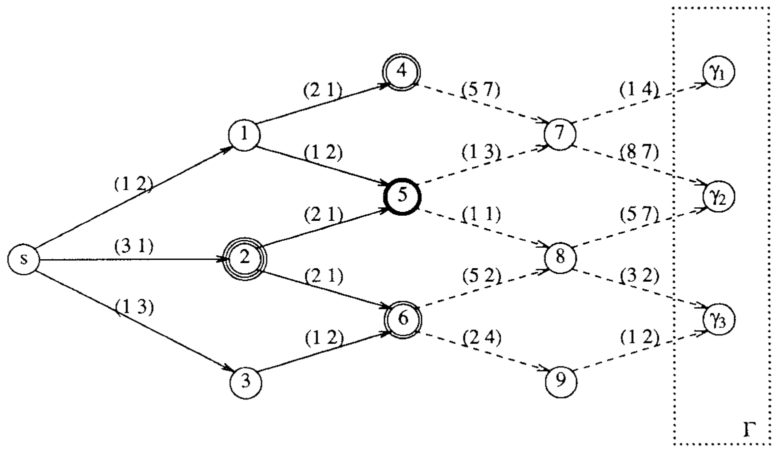


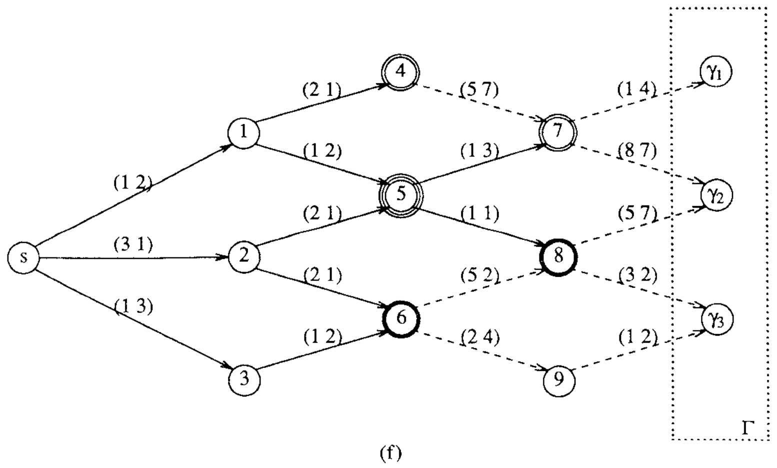
FIG. 3. Graphical trace of the operation of *MOA** on Example 1. (a) Iteration 0 (b) Iteration 1. (c) Iteration 2. (d) Iteration 3. (e) Iteration 4. (f) Iteration 5. (g) Iteration 6. (h) Iteration 7. (i) Iteration 8. (j) Iteration 9. (k) Iteration 10.



(d)



(e)



(f)

FIG. 3. Graphical trace of the operations of *MOA** on Example 1 (continued).

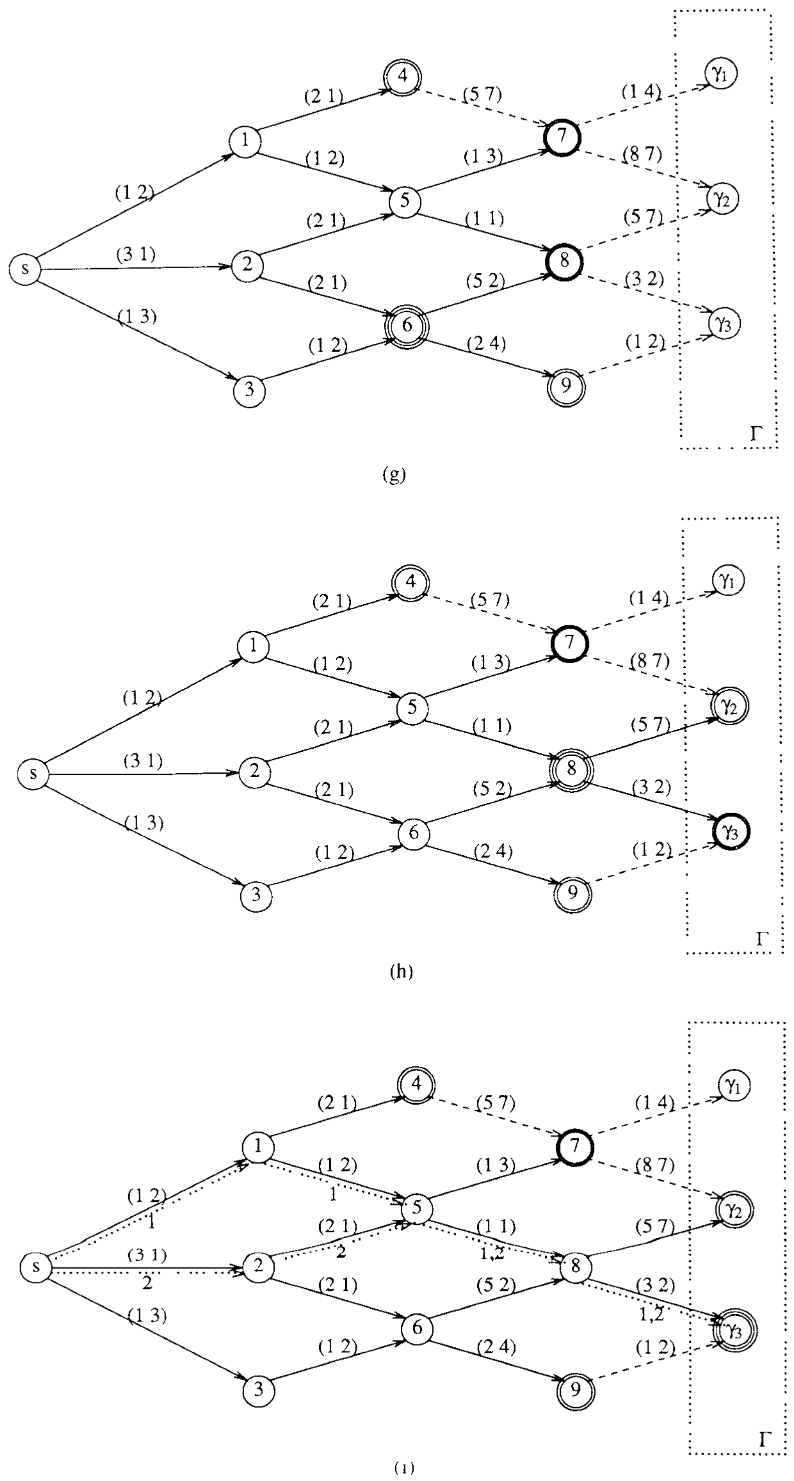
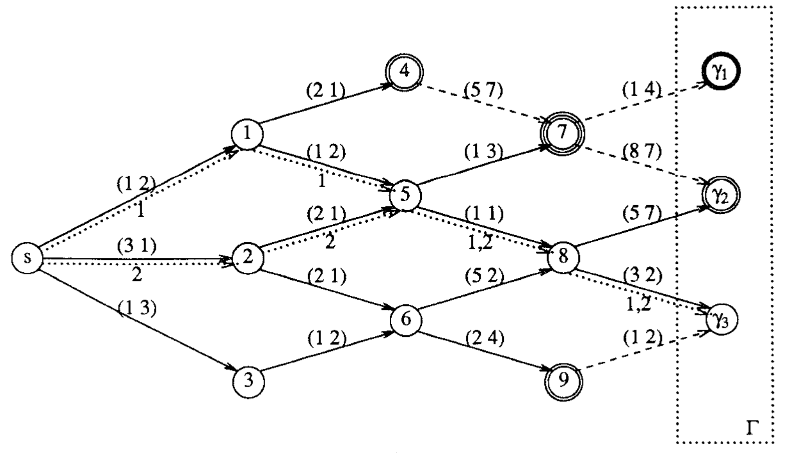
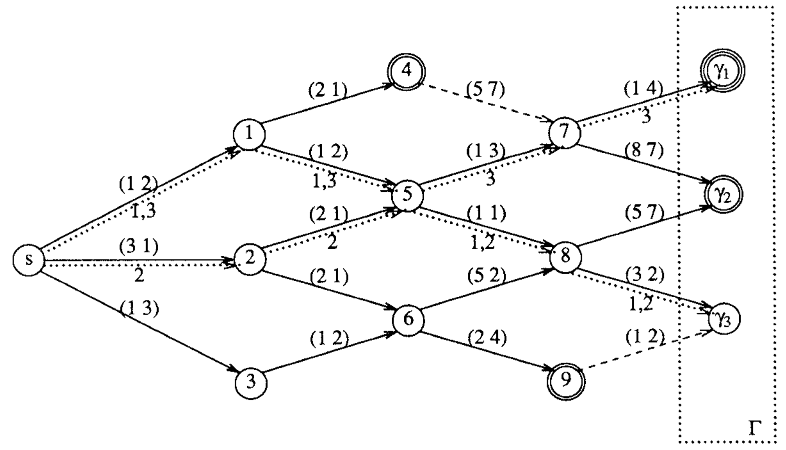


FIG. 3. Graphical trace of the operations of *MOA** on Example 1 (continued).



(j)



(k)

FIG. 3. Graphical trace of the operations of *MOA** on Example 1 (continued).

the figures. Final solution paths are shown as additional numbered dotted lines in the graphs. Nodes of the initial problem state-space graph are shown as circles. *OPEN* nodes are shown with double circles. Nondominated members of *OPEN*; that is, nodes of *ND*, are shown with large darkened circles. The node selected for expansion in each iteration is depicted with three circles. Finally, *CLOSED* nodes may be identified by the fact that they are single solid circles for which at least one incoming arc is no longer dashed.

Table II traces the evolution of the critical quantities and sets as the search progresses. Elements of the sets *G*, *F*, and *LABEL* (see definitions in Section 3.3) are only shown if they have been altered in an iteration. Nodes in *ND* (see definition in Section 3.3) are marked with an asterisk in the second column and the node selected for expansion is marked with a second asterisk in the same column. Nodes were selected for expansion from *ND* arbitrarily. Specific heuristic selection rules should be developed for particular domains of application.

TABLE II. NUMERICAL TRACE OF THE OPERATION OF MOA* ON EXAMPLE 1.

	$OPEN_k$			Backpointer Adjustments			$SOLUTION\_COSTS_k$ and
k	n	New $G_k(n)$	New $F_k(n)$	n'	$LABEL_k(n', n)$	$CLOSED_k$	$GOALS_k$
0	s^{**}	(0 0)	(1 2)(3 1)			$\varnothing$	$\varnothing$
1	1^{**}	(1 2)	(2 4)(3 3)	s	(1 2)	s	$\varnothing$
	2^{*}	(3 1)	(5 2)	s	(3 1)		
	3	(1 3)	(2 5)	s	(1 3)		
2	$2^{*}, 3^{**}$					$s, 1$	$\varnothing$
	4	(3 3)	(8 10)	1	(3 3)		
	5	(2 4)	(3 5)	1	(2 4)		
3	$2^{**}, 4, 5^{*}$					$s, 1, 3$	$\varnothing$
	6	(2 5)	(7 7)(4 9)	3	(2 5)		
4	4					$s, 1, 2, 3$	$\varnothing$
	5^{**}	(2 4)(5 2)	(3 5)(6 3)	2	(5 2)		
	6	(2 5)(5 2)	(4 9)(7 6) (10 4)	2	(5 2)		
5	$4, 6^{**}$					$s, 1, 3, 2, 5$	$\varnothing$
	7	(3 7)(6 5)	(4 11)(7 9)	5	(3 7)(6 5)		
	8^{*}	(3 5)(6 3)	(6 7)(9 5)	5	(3 5)(6 3)		
6	$4, 7^{*}, 8^{**}$					$s, 1, 3, 2, 5, 6$	$\varnothing$
	9	(7 6)(4 9)	(8 8)(5 11)	6	(7 6)(4 9)		
7	$4, 7^{*}, 9$					$s, 1, 3, 2, 5, 6$	$\varnothing$
	γ_2	(8 12) (11 10)	(8 12) (11 10)	8	(8 12)(11 10)		
	γ_3^{**}	(6 7)(9 5)	(6 7)(9 5)	8	(6 7)(9 5)		
8	$4, 7^{**}, 9, \gamma_2$					$s, 1, 3, 2, 5, 6, 8, \gamma_3$	$\{(6\ 7), (9\ 5)\}$; $\{\gamma_3\}$
9	$4, 9, \gamma_2$					$s, 1, 3, 2, 5, 6, 8, \gamma_3, 7$	$\{(6\ 7), (9\ 5)\}$, $\{\gamma_3\}$
	γ_1^{**}	(4 11)(7 9)	(4 11)(7 9)	7	(4 11)(7 9)		
10	4, 9					$s, 1, 3, 2, 5, 6, 8, \gamma_3, 7, \gamma_1$	$\{(6\ 7), (9\ 5), (4\ 11)\}$, $\{\gamma_3, \gamma_1\}$
	γ_2						

The search ends in Step (2) of the 11th iteration after finding the following three nondominated solution paths and their associated costs:

$$P_1^* = \{s, 1, 5, 8, \gamma_3\} \qquad c(P_1^*) = (6\ 7)$$
$$P_2^* = \{s, 2, 5, 8, \gamma_3\} \qquad c(P_2^*) = (9\ 5)$$
$$P_3^* = \{s, 1, 5, 7, \gamma_1\} \qquad c(P_3^*) = (4\ 11).$$

3. *Multiobjective Search Problem Definition*

The example above was an instance of the class of multiobjective search problems defined in this section. We begin with an abstract formulation of the general problem class being addressed, followed by the definition of dominance used in this paper. Some notation and terminology is considered next and we conclude with several assumptions.

3.1. Problem Formulation. The class of problems considered here differs from the corresponding single-objective version only in its cost structure and solution definition. The definition of a *solution path* is identical; however, any path in **G** from the start node to a member of the goal set. We formulate our problem as shown in Figure 4. Note that several distinct paths may have the same nondominated costs and may terminate in the same or different goal nodes. All such equivalent paths must be produced in the current problem formulation.

3.2. Definition of Dominance. Throughout this paper, we assume that all objectives are formulated in terms of some quantity to be minimized (see Assumption 4). The following definition of dominance reflects this assumption. Maximization problems may be handled with slight modifications.

Definitions. Given a set Y of M-vectors, we define a relation $R \subseteq Y \times Y$, as follows:

$$(y, y') \in R \Leftrightarrow y_i \leq y_i' \forall i \in 1, \dots, M.$$

This relation induces a strict partial order R' on Y in the following standard fashion:

$$(y, y') \in R' \Leftrightarrow (y, y') \in R \text{ and } (y', y) \notin R.$$

An alternative characterization of this induced relation is as follows:

$$(y, y') \in R' \Leftrightarrow y_i \leq y_i' \forall i \in 1, \dots, M \text{ and } y \neq y'.$$

We say that y *dominates* y' (in Y) if $(y, y') \in R'$. An element $y \in Y$ is said to be *nondominated* (in Y) if there does not exist another element $y' \in Y$ such that y' dominates y (in Y). We use the notation *nondom* $\{Y\}$ to indicate the nondominated subset of Y; that is,

$$nondom\{Y\} = \{y \in Y: \nexists y' \in (y', y) \in R'\}.$$

This dominance relation provides our notion of preference among paths through the use of path costs and cost estimates. We say that a path P is nondominated in a set of paths Q when a vector evaluation of P is nondominated in the set of vector evaluations associated with the elements of Q.

3.3. Notation and Terminology. The following notation and terminology is useful for formalizing the MOA^* algorithm. To the extent allowed by the different requirements of the multiobjective formulation, the notation below has been defined in a manner consistent with that used for A^* by [22], [23], and [24]. Some terminology is more appropriately defined in the context of its use, so that other definitions appear throughout the paper. Iteration subscripts that appear on the sets *OPEN*, *ND*, *CLOSED*, *SOLUTION_COSTS*, *SOLUTION_GOALS*, *G*, *LABEL*, and *F* make it easier to provide clear proofs of the properties of the algorithm. The subscripts are not required for the

Given:

—A problem state-space, represented as a locally-finite directed graph; states are nodes in the graph.
—A single start node in the graph.
—A finite set of goal nodes in the graph.
—A positive, vector-valued cost associated with each arc in the graph.
—A set of vector-valued heuristic functions providing information on costs to be accrued along a portion of a solution path.
—A notion of preference among paths based on their costs.

Find:

—The set of all nondominated solution paths in the graph.

FIG. 4. Multiobjective Search Problem Formulation

correct operation of MOA^* and should be dropped from any efficient software implementation of the procedure.

$\mathbf{G} = (N(\mathbf{G}), A(\mathbf{G}))$
—*State-space graph* representing the problem to be solved. The detailed correspondence between the problem and its graphical representation is highly problem-specific. $N(\mathbf{G})$ is a countable set of *nodes*. $A(\mathbf{G}) \subseteq N \times N$ represents the set of *directed arcs* connecting pairs of nodes. For any arc $\alpha = (n, n')$, node n' is referred to as the *head* of α and n as the *tail* of α. For any two graphs $\mathbf{G}'$ and $\mathbf{G}''$, we say that $\mathbf{G}'$ is contained in $\mathbf{G}''$, written $\mathbf{G}' \subseteq \mathbf{G}''$, if $A(\mathbf{G}') \subseteq A(\mathbf{G}'')$ and $N(\mathbf{G}') \subseteq N(\mathbf{G}'')$.

$s \in N(\mathbf{G})$
—Unique *start node* in $\mathbf{G}$.

$\Gamma \subseteq N(\mathbf{G})$
—Finite, nonempty set of *goal nodes*.

$P = (n_0, n_1, n_2, \ldots)$
—*Path* in some graph $\mathbf{G}'$. P consists of the sequence of two or more nodes, $n_i \in N(\mathbf{G}')$, and associated arcs, $a_i = (n_i, n_{i+1}) \in A(\mathbf{G}') \forall i = 1, 2, 3, \ldots$ n_0 is said to be the *root* of path P. We use $N(P)$ to represent the set of nodes on P and $A(P)$ to indicate the set of arcs on P. We allow at most one node, n_i, on P to have the property that the arc $(n_i, n') \notin A(P) \forall (n_i, n') \in A(G')$, and we say that n_i is the *terminal* node of P or the *last* node of P. The *length* of P, indicated by *length* $\{P\}$, is the number of arcs on P and if *length* $\{P\}$ is not finite we say that P is an *infinite path*. We require a path to have length at least 1 (at least 2 nodes and 1 arc) because a trivial path from a node to itself will never be relevant in the problems we address. A *subpath* of P is any path $P' = (N(P'), A(P'))$ such that $N(P') \subseteq N(P)$ and $A(P') \subseteq A(P)$. P' is said to be a *proper subpath* of P if it is a subpath of P such that $A(P') \subset A(P)$. A path is said to be *cyclic* if it contains a subpath, the root and tail of which are the same node. A path that is not cyclic is said to be *acyclic*. For any P' and P'' such that the last node of P' is the same as the root of P'', we write $P = (P', P'')$ to mean the path formed by the concatenation of P' and P'', in which case

$$P = (P', P'') \Leftrightarrow N(P) = N(P') \cup N(P'') \text{ and } A(P) = A(P') \cup A(P'').$$

We say a path P is contained in a graph $\mathbf{G}'$, written $P \subseteq \mathbf{G}'$, if $A(P) \subseteq A(\mathbf{G}')$, which also implies that $N(P) \subseteq N(\mathbf{G}')$.

$\mathbf{P}(X_1, X_2) = \{P$: the root of P is $x_1 \in X_1$, and the tail of P is $x_2 \in X_2\}$

—Set of all acyclic *paths* in $\mathbf{G}$ having their root in the finite set of nodes X_1 and their terminal node in the finite node set X_2. If either set of nodes is a singleton, for example $X_1 = \{n\}$, we abbreviate by omitting the set notation as follows: $\mathbf{P}(X_1, X_2) = \mathbf{P}(\{n\}, X_2) = \mathbf{P}(n, X_2)$. We indicate the set of nodes, each of which is on a path in $\mathbf{P}(X_1, X_2)$, with the notation $N(\mathbf{P}(X_1, X_2))$. We define the intersection of a graph $\mathbf{G}'$ and a set of paths $\mathbf{P}$ as follows:

$$\mathbf{G}' \cap \mathbf{P} = \{P \in \mathbf{P}\colon P \subseteq \mathbf{G}'\}.$$

Let X_1 and X_2 be sets of nodes in $\mathbf{G}$. If $N(P) \cap X_1 = \{n\}$ and $N(P) \cap X_2 = \{n'\}$ for some path P, then we define the intersection of P and $\mathbf{P}(X_1, X_2)$ as follows:

$$P \cap \mathbf{P}(X_1, X_2) = P',$$

where P' is the subpath of P between nodes n and n', and $P' \in \mathbf{P}(X_1, X_2)$. If no such P' exists, then we leave the intersection of P and $\mathbf{P}(X_1, X_2)$ undefined; however, we only use this form in cases for which the result is well defined. The set of all paths in $\mathbf{G}$, including those that are infinite and/or cyclic, is denoted $\mathbf{P}'$. The set of all acyclic paths in $\mathbf{G}$ is indicated by $\mathbf{P} \subseteq P'$.

$\mathbf{P}^*(X_1, X_2) = nondom\{\mathbf{P}(X_1, X_2)\}$

—Set of all *nondominated paths* from any node in X_1 to any member of node set X_2.

$\Gamma^* \subseteq \Gamma$

—Set of *nondominated goal nodes*; that is, the set of goal nodes that can be reached from s via one or more nondominated solution paths in $\mathbf{G}$. The set of nondominated solution paths is denoted by $\mathbf{P}^*(s, \Gamma)$. Formally,

$$\Gamma^* = \{\gamma \in \Gamma\colon \mathbf{P}(s, \gamma) \cap \mathbf{P}^*(s, \Gamma) \neq \varnothing\}.$$

$c(P) = \{c_1(P), \ldots, c_M(P)\}$

—Vector-valued *path cost* associated with path P, where M is a finite, positive integer corresponding to the number of objectives under consideration. In a slight abuse of notation, for any set of paths $\mathbf{P}_1$, we use $c(\mathbf{P}_1) = \{c(P)\colon P \in \mathbf{P}_1\}$.

$C^* = \{c^*(P)\} = \{c(P)\colon P \in \mathbf{P}^*(s, \Gamma)\}$

—Set of all *costs of nondominated solution paths.*

$OPEN_k \subseteq N(\mathbf{G})$

—Set of nodes that are considered as *potential candidates for expansion* by MOA^* at iteration k. If $n \in OPEN_k$, but $n \notin OPEN_{k_1} \forall k_1 < k$ then we say that n was *opened* during iteration $k - 1$. If n was opened during some iteration k_1, $n \notin OPEN_{k_2}$ for some $k_2 > k_1$, and $n \in OPEN_k$ for some $k > k_2$, then we say that n was *reopened* during iteration $k - 1$.

$SCS(n) \subseteq N(\mathbf{G})$

—Set of all nodes that are *successors* or *children* of node n; that is, $SCS(n) = \{n'\colon (n, n') \in A(\mathbf{G})\}$. Node n is referred to as a *parent* or *predecessor* of each $n' \in SCS(n)$. We indicate the set of all such parents of node n by $SCS^{-1}(n)$.

$CLOSED_k \subseteq N(\mathbf{G})$

—Set used in MOA^* to keep track of nodes that have been generated, but are not on $OPEN_k$. Node $n \in N(\mathbf{G})$ is said to be *generated* during iteration k if its representation is computed from that of one of its parents. A node is said to be *expanded* when all its successors have been generated. An arc is said to be *traversed* if the node at its head has been generated from the node at its tail. Note that according to these definitions and the construction of the algorithm, $OPEN_k \cap CLOSED_k = \emptyset \forall k$.

$LABEL_k(n', n)$

—Set of *accrued costs* of paths from s through n to n' (where the cost of arc (n, n') is included in each such path cost) that are nondominated among all such paths discovered through iteration $k - 1$. We say there exists a *backpointer* from n' to n or a backpointer for arc $\alpha = (n, n')$ at iteration k if and only if $LABEL_k(n', n) \neq \emptyset$. $LABEL_k(n', n)$ is said to be the *label set* of arc $\alpha = (n, n')$ at iteration k. We also write $LABEL_k(\alpha)$ for this set.

$TSG_k \subseteq \mathbf{G}$

—*Traversal subgraph* at iteration k, consisting of the arcs that have backpointers associated with them, and the nodes associated with those arcs, at the start of iteration k, with the arcs of TSG_k directed opposite to the backpointers; that is, directed consistent with the arcs of the underlying graph. Note that the construction of MOA^* ensures that $OPEN_k \subseteq N(TSG_k)$. A path is said to be *discovered* during iteration k if it is in TSG_{k+1}, but not in TSG_i for any $i \leq k$. We say that a path is *rediscovered* during iteration k if it has been discovered during some iteration $i < k$ and it is in TSG_{k+1}, but not in TSG_k.

$ESG_k \subseteq \mathbf{G}$

—*Explicated subgraph* at iteration k, defined as follows:

$$ESG_k = \big(N(ESG_k), A(ESG_k)\big),$$

where $N(ESG_k) = \{\cup N(TSG_i): 0 \leq i < k\}$ and $A(ESG_k) = \{$all arcs that have been traversed prior to iteration $k\}$. As [24, p. 75] has pointed out for scalar problems, a path can be in ESG_k and not in any of the traversal subgraphs that make up ESG_k.

$SOLUTION_k \subseteq \mathbf{P}(s, \Gamma)$

—Collection of *solution paths* that have been discovered prior to the start of iteration k. Note that this set is maintained implicitly encoded using *SOLUTION_COSTS* and *SOLUTION_GOALS*, defined below, until the last step before the algorithm terminates.

$SOLUTION\_GOALS_k \subseteq \Gamma$

Set of *nondominated goals* associated with paths in $SOLUTION_k$.

$G_k(n) = nondom\ \{\cup LABEL_k(n, n'): n' \in SCS^{-1}(n)\}$

—Set of *nondominated accrued costs* for node n at iteration k. $G_k(n)$ is the set of cost vectors for all paths from s to n that are nondominated among the set of all paths from s to n that have been discovered prior to the start of iteration k. We define $G_k(s) = \{0\} \forall k$.

$SOLUTION\_COSTS_k$

—Set of *nondominated solution path costs* associated with goals in $SOLUTION\_GOALS_k$. Formally,

$$\begin{aligned} SOLUTION\_COSTS_k &= nondom\{c(P) \colon P \in SOLUTION_k\} \\ &= nondom\{\cup G_k(\gamma) \colon \gamma \in SOLUTION\_GOALS_k\}. \end{aligned}$$

$G^*(n)$

—Set of all *nondominated path costs* between the start node and node n. Thus,

$$G^*(n) = \{c(P) \colon P \in \mathbf{P}^*(s, n)\} = nondom\{c(P) \colon P \in \mathbf{P}(s, n)\}.$$

$H^*(n)$

—Set of all *nondominated path costs* associated with a portion of at least one nondominated solution path; specifically, the cost of the portion between n and Γ. Formally,

$$H^*(n) = \{c(P) \colon P \in \mathbf{P}^*(n, \Gamma)\} = nondom\{c(P) \colon P \in \mathbf{P}(n, \Gamma)\}.$$

$h \colon N(\mathbf{G}) \rightarrow \Re^{M++}$

—*Heuristic function*, where the values $h(n)$ are intended to approximate some member of the set of actual nondominated costs defined by $H^*(n)$ above. We use $\Re^{M++}$ to indicate the set $\{x \colon x \in \Re^M, x \geq 0\} \cup \{\infty\}$. H is the set of all such heuristic functions specified for a given problem. $H(n)$ is the nondominated subset of the collection of heuristic function values (vectors) associated with node n; that is

$$H(n) = nondom\{h(n) \colon h \in H \text{ and } n \in N(\mathbf{G})\}.$$

$PP_k(n, n')$

—*Pointer path* from node n to node n', indicating a path in $\mathbf{G}$ from n' to n that is currently nondominated with respect to the set of accrued costs of discovered paths from n' to n. Formally, a pointer path is said to exist between nodes n and n' if there is a path $P \in TSG_k \cap \mathbf{P}(n', n)$ such that:

$$\forall (n'', n''') \in A(P) \exists c' \in LABEL(n''', n'') \cap G_k(n'''),$$

and, for the most common case where $n'' \neq s$, we also must have

$$\exists c'' \in G_k(n'') \ni c' = c(n'', n''') + c''.$$

At iteration k, we denote the set of all such pointer paths between n and any ancestor n' of n by $\mathbf{PP}_k(n, n')$.

$F_k(n)$

—Set of *node selection values* for node n at iteration k. This set is defined as follows:

$$\begin{aligned} F_k(n) &= \varnothing && \text{if } n \notin ESG_k \\ &= nondom\{g + h \colon g \in G_k(n) \text{ and } h \in H(n)\} && \text{if } n \in ESG_k. \end{aligned}$$

F_k

—Set of node selection values associated with nodes on $OPEN_k$; that is,

$$F_k = \{f \in F_k(n) : n \in OPEN_k\}.$$

C_k

—Set of costs to be used for making selections for expansion from $OPEN_k$, consisting of node selection values for nodes in $OPEN_k$ and costs of solution paths that have already been discovered and placed in $SOLUTION\_COSTS_k$. Formally,

$$C_k = F_k \cup SOLUTION\_COSTS_k.$$

$F^*(n)$

—Set of costs of paths that are nondominated among all paths from s through n to Γ. According to this definition and the others above

$$C^* = F^*(s) = H^*(s) = nondom\{G^*(n) : n \in \Gamma^*\}.$$

Formally,

$$F^*(n) = nondom\{g + h : g \in G^*(n), h \in H^*(n)\}.$$

$ND_k \subseteq OPEN_k$

—Set of all nodes in $OPEN_k$ that have node selection function values that are not dominated by:

(1) the cost of any solution path discovered prior to iteration k (i.e., in $SOLUTION\_COSTS_k$), nor by

(2) the node selection function values of other potential solutions represented by nodes on $OPEN_k$.

Formally,

$$ND_k = \{n \in OPEN_k : F_k(n) \cap nondom\{C_k\} \neq \emptyset\}.$$

3.4. Assumptions. The following assumptions are made throughout the paper. When $M = 1$ these assumptions reduce to those that are usually made for the scalar case; that is, the case of A^*.

Assumption 1. $\mathbf{G}$ is locally finite; that is, $SCS(n)$ and $SCS^{-1}(n)$ are finite sets for all $n \subseteq N(\mathbf{G})$.

Assumption 2. Path costs are additive; that is, the cost of path P is equal to the vector sum of the costs of the arcs along P; that is,

$$c(P) = \sum_{\alpha \in A(P)} c(\alpha).$$

The next two assumptions guarantee that infinite paths may be detected in a finite number of iterations of the search algorithm.

Assumption 3. All cost vectors are positive and uniformly bounded in all components. That is, there exist δ and $B1$, independent of m and α, such that

$$0 < \delta \leq c_m(\alpha) \leq B1 < \infty \forall \alpha \in A(\mathbf{G}) \text{ and } m \in \{1, \ldots, M\}.$$

Assumption 3 ensures that actual costs of finite paths always dominate costs of infinite paths. Let $\setminus$ represent the set difference operator; that is, for two sets A and B define $A \setminus B = \{x\colon\ x \in A \text{ and } x \notin B\}$. As usual, let * represent the multiplication operator. In view of Assumption 3, the following assumptions concerning valid heuristic functions are clearly reasonable.

Assumption 4. The heuristic functions satisfy the following conditions:

$$h_i \geq 0 \forall n \in N(\mathbf{G}) \setminus \Gamma,\ h \in H(n), \text{ and } i \in \{1, \ldots, M\},$$
$$h_i = 0 \forall h \in H(\gamma),\ \gamma \in \Gamma, \text{ and } i \in \{1, \ldots, M\},$$

and there exists a finite positive integer $B2$, independent of i and n', such that, for any $n \in N(\mathbf{G})$ with $\mathbf{P}(n, \Gamma) \neq \varnothing$, if $P \in \mathbf{P}^*(n, \Gamma)$, then

$$\forall n' \in N(P) \exists h \in H(n') \ni h_i(n') \leq B2^* \mathit{length}\{P\} \forall i \in \{1, \ldots, M\}.$$

The next three assumptions are made merely for convenience in the presentation of our results. Assumption 5 was discussed above. As long as there is only a finite number of cost vectors associated with each pair (head, tail), Assumption 6 can be relaxed and all the results below will still hold. Assumption 7 is made without loss of generality in view of Assumption 5. Note that Assumption 7 is an assumption about the paths that we are considering in the presentation below; it is not a restriction on the underlying structure of the search graph.

Assumption 5. All objectives are expressed as minimizations of some component of the additive cost function, c.

Assumption 6. No two arcs in $\mathbf{G}$ may have the same head and tail.

Assumption 7. All paths are acyclic.

4. *Formal Properties of MOA**

In the following sections, we show that MOA^* has the same useful properties that A^* has been shown to possess. In the first section, we discuss a multiobjective form of Bellman's Principle of Optimality and present some preliminary results. The second section contains results concerning termination and completeness of MOA^*. The admissibility of MOA^* is proved in the third section, followed in the fourth section by some results concerning comparison of heuristics. The final section introduces the concepts of monotonicity and consistency with respect to heuristic functions and presents several associated results.

4.1. The Principle of Optimality and Other Preliminary Results.

In this section, we state a version of the Principle of Optimality that applies in our current setting, and prove several results that are needed in subsequent sections. We begin with some basic definitions and several simple results.

Many graph theorists define a graph to consist of a finite set of nodes and a finite set of arcs; see, for example, [2, p. 2], [9, p. 9], and [28, p. 3]. Consistent with earlier researchers in heuristic search theory, we adopt the more general approach of [27, p. 14] and [20, p. 168] and allow for the possibility of infinite graphs; see also Assumption 1 above. Thus, the following definitions are necessary.

Definition. A graph is said to be *finite* if it contains a finite number of arcs, and thus, a finite number of nonisolated nodes.

Definition. A graph is said to be *infinite* if it is not finite.

Our first results are simple, yet they deserve explication due to the possible confusion introduced by our allowance for infinite graphs. A key implication of our first lemma is that every solution path is finite.

LEMMA 1. *If $n, n' \in N(\mathbf{G})$ and $P \in \mathbf{P}(n, n')$, then P is a finite path.*

PROOF. Let $n \in N(\mathbf{G})$ and let P be a path with n as its root. By the definition of an infinite path, P cannot have a terminal node, and, specifically, n' will not be a terminal node of P. □

Definitions. Let n_i and n_j be two nodes on a path P along which nodes are indexed as follows: $P = (n_0, n_1, n_2, \ldots)$. Then n_i is said to be *deeper than* n_j if $i > j$ and *shallower than* n_j if $i < j$. If n_i is deeper than n_j, then n_i is said to be a *descendant* of n_j, and n_j is said to be an *ancestor* of n_i.

In order to more easily discuss the properties of MOA^*, we introduce some additional terminology related to the operation of the algorithm. Whenever Step (2.2.1) of the algorithm is executed, we say the node in question has been *selected for expansion*. In view of the remainder of Step (2.2), we have the following obvious fact:

Fact 1. If a node n is selected for expansion during iteration k, then $n \in ND_k \cap CLOSED_{k+1}$.

Nodes are expanded in Step (5) of the algorithm. Therefore, we also observe the following obvious fact based on the algorithm structure:

Fact 2. If n is not a goal node and n is selected for expansion during iteration k, then n is expanded during iteration k.

The next lemma is a chain of four simple results concerning membership in the sets *OPEN* and *CLOSED*, each dependent on the one before it.

LEMMA 2

(*a*) *Node n is first generated during iteration k if and only if k is the smallest integer such that*:

$$n \in OPEN_{k'} \cup CLOSED_{k'} \forall k' > k.$$

(*b*) *If $n' \in SCS(n)$ and $n \in CLOSED_k$, then $n' \in OPEN_k \cup CLOSED_k$.*

(*c*) *Let P be any path in $\mathbf{G}$ having $n_0 \in CLOSED_k$ as its root, and let*

$$n_{i+1} = SCS(n_i) \cap N(P) \forall n_i \in N(P).$$

Then, either $n_i \in CLOSED_k \forall n_i \in N(P)$ or $N(P) \cap OPEN_k \neq \emptyset$.

(*d*) *If $n \in N(\mathbf{G})$, then for all $P \in \mathbf{P}(s, n)$ and $k \geq 0$, either $N(P) \subseteq CLOSED_k$ or $N(P) \cap OPEN_k \neq \emptyset$.*

PROOF

(a) Note that the construction of the sets *OPEN* and *CLOSED* guarantees that

$$OPEN_k \cup CLOSED_k \subseteq OPEN_{k+1} \cup CLOSED_{k+1} \forall k. \qquad (1)$$

Assume node n was generated during iteration $k < k'$. Then, by Step (5) of the algorithm,

$$n \in OPEN_{k+1} \cup CLOSED_{k+1} \subseteq OPEN_{k'} \cup CLOSED_{k'},$$

where the second set inclusion follows from (1).

To see the converse, note that nodes enter *CLOSED* only from *OPEN* (Step (2)). Nodes enter *OPEN* from *CLOSED* (Step (5)(d) or by being generated for the first time (Step (5)(c)). Therefore, the only way that nodes may enter *OPEN* ∪ *CLOSED* is via node generation in Step (5)(c). So if k is the smallest integer such that $n \in OPEN_{k'} \cup CLOSED_{k'} \forall k' > k$, then k must be the first iteration in which n is generated.

(b) Nodes enter *CLOSED* only when they are selected for expansion (Step (2)). Since $n \in CLOSED_k$, Fact 1 implies that all members of $SCS(n)$ have been generated at some iteration $k' < k$. Therefore, by part (a), $n' \in OPEN_k \cup CLOSED_k$.

(c) By assumption,

$$n_0 \in CLOSED_k \subseteq OPEN_k \cup CLOSED_k.$$

Assume $n_i \in OPEN_k \cup CLOSED_k$. If $n_i \in OPEN_k$, then the result holds. Assume $n_i \in CLOSED_k$. By part (b), $n_{i+1} \in OPEN_k \cup CLOSED_k$. The result follows by complete induction.

(d) Follows directly from part (c) and the operation of the algorithm, which ensures that $s \in CLOSED_k \forall k > 0$. □

The next result ensures that the nondominated set of paths will be finite in all cases of interest.

LEMMA 3. *Let $n, n' \in N(\mathbf{G})$. If $\mathbf{P}(n, n') \neq \varnothing$, then $\mathbf{P}^*(n, n')$ is finite.*

PROOF. Let $n, n' \in N(\mathbf{G})$, assume $\mathbf{P}(n, n') \neq \varnothing$, and define

$$T^k(n, n') = \{P \in \mathbf{P}(n, n'): \mathit{length}\,\{P\} \leq k\};$$
$$T(n, n') = \{\cup T^k(n, n'): 1 \leq k \leq \infty\}.$$

We wish to show that *nondom* $\{T(n, n')\}$ is finite. By Assumption 1, $T^k(n, n')$ is a finite set for any finite positive integer k. Let P be a finite path from n to n'; at least one such path exists by Lemma 1 and the assumption $\mathbf{P}(n, n') \neq \varnothing$. Let δ and B be selected as described in Assumption 3; that is, so that

$$0 < \delta \leq c_m(\alpha) \leq B < \infty \forall \alpha \in A(\mathbf{G}),\ m \in \{1, \ldots, M\}.$$

This clearly implies that

$$c_m(P) \leq \mathit{length}\{P\}^* B \forall m \in \{1, \ldots, M\}.$$

Now let $K = \mathit{length}\,\{P\}^* B/\delta$ so that

$$\{c(P), c(P')\} \in R' \forall P' \ni \mathit{length}\,\{P'\} > K.$$

Therefore,

$$\mathit{nondom}\{T(n, n')\} \subseteq T^K(n, n'),$$

and the result is proved. □

COROLLARY 1. *The sets $H^*(n)$, $G^*(n)$, $F^*(n)$, $\mathbf{P}^*(s, \Gamma)$, and C^* are finite.*

PROOF. Immediate from the definitions of the indicated sets and Lemma 3. □

We now introduce the following well-known result, which immediately implies Lemma 5 and its corollary. For an example of the similar, but more general, decision theoretic result, see [30, p. 21].

LEMMA 4. *Given a finite nonempty set S and a partial order defined using the reflexive and transitive binary relation $R \subseteq S \times S$, the nondominated subset of S with respect to R is nonempty.*

LEMMA 5. *Let $n, n' \in N(\mathbf{G})$. If $\mathbf{P}(n, n') \neq \emptyset$, then $\mathbf{P}^*(n, n') \neq \emptyset$.*

PROOF. Immediate from Lemma 4. □

COROLLARY 2. *If $\mathbf{P}(s, \Gamma) \neq \emptyset$, then $\mathbf{P}^*(s, \Gamma) \neq \emptyset$.*

PROOF. Immediate from Lemma 5. □

The following property is fundamental to the remainder of our development.

Definition. A cost and preference structure for a path problem is said to be *order-preserving* if, for all paths P_i in $\mathbf{G}$, $i = 1, 2, 3, 4, 5$ such that (P_1, P_3), (P_2, P_3), (P_3, P_4), and (P_3, P_5) are also paths in G, the following conditions hold:

$$\{P_1, P_2\} \in R \Rightarrow \{(P_1, P_3), (P_2, P_3)\} \in R$$
$$\{P_4, P_5\} \in R \Rightarrow \{(P_3, P_4), (P_3, P_5)\} \in R.$$

Our additive cost assumption and simple preference structure based on the standard vector order relation clearly satisfy these conditions. The next lemma provides an extremely useful implication of this property.

LEMMA 6 (*Principle of Optimality*). *Let $n, n' \in N(\mathbf{G})$. If $\mathbf{P}(n, n') \neq \emptyset$, then, for any $P \in \mathbf{P}^*(n, n')$ and $n'' \in N(P)$, the following two conditions hold*:

$$P \cap \mathbf{P}(n, n'') \in \mathbf{P}^*(n, n'')$$

and

$$P \cap \mathbf{P}(n'', n') \in \mathbf{P}^*(n'', n').$$

PROOF. First note that the assumption $\mathbf{P}(n, n') \neq \emptyset$ together with Lemma 5 implies that $\mathbf{P}^*(n, n') \neq \emptyset$. Let $P \in \mathbf{P}^*(n, n')$, $n'' \in N(P)$ and assume $P \cap \mathbf{P}(n, n'') \notin \mathbf{P}^*(n, n'')$. If no such $n'' \notin \{n, n'\}$ exists, then the results hold trivially, so assume that $n'' \notin \{n, n'\}$. According to our definitions of $\mathbf{P}^*(n, n'')$ and R',

$$\exists P' \in \mathbf{P}^*(n, n'') \ni (c(P'), c(P \cap \mathbf{P}(n, n''))) \in R'.$$

By the transitivity of our strict partial preference order R', this implies that

$$(c(P', P \cap \mathbf{P}(n'', n')), c(P)) \in R',$$

which contradicts the assumption in the lemma statement that $P \in \mathbf{P}^*(n, n')$. The other half of the proof is similar. □

Lemma 6 is a form of Bellman's Principle of Optimality [3]. Equivalent forms of this principle that are relevant in various multiobjective settings have been discussed by [6], [7], [12], and [25]. For another approach to the use of heuristic search techniques for multiobjective problems in which this form of the Principle of Optimality does not hold, see [32]. The version of the Principle of Optimality in Lemma 6 will be important in the proofs of several results below. We now turn to the issues of termination and completeness of *MOA**.

4.2. Termination and Completeness. We show that *MOA** terminates using the following definition.

Definition. An algorithm is said to *terminate* if it is guaranteed to halt after a finite number of iterations.

In *MOA**, the termination condition is met when Step (2) is reached and $ND = \varnothing$ (Step (2.1)). The term "algorithm" is sometimes defined as a procedure that is guaranteed to terminate; see, for example, [1, p. 2]. If we were to adopt such a definition, then the following termination proofs would show that *MOA** qualifies as an algorithm in this restricted sense.

Lemma 7. *If* $\mathbf{P}(s, \Gamma) \neq \varnothing$, *then only a finite set of nodes may become members of ND; that is, only a finite number of nodes are candidates for expansion.*

Proof. Let $P \in \mathbf{P}(s, \Gamma)$. By Lemma 1, P is finite. We consider two cases. For the first case, assume there is some iteration k for which $N(P) \subseteq CLOSED_k$. By the construction of *MOA**, $c(P) \in SOLUTION\_COSTS_k$. By Assumption 3, there is some positive real number δ and a finite positive integer B, such that

$$(c(P), c(P')) \in R' \forall P' \ni length\,\{P'\} \geq K,$$

where $K = length\,\{P\}{*}B/\delta$. Define T^k as follows:

$$T^k = \{\cup N(P) \colon P \in \mathbf{P}(s, n),\ n \in N(\mathbf{G}), \text{and } length\,\{P\} \leq k\}.$$

By the definition of $G_k(n)$,

$$\forall g \in G_k(n) \exists P'' \in \mathbf{P}(s, n) \ni g = c(P'').$$

Therefore, if $n \notin T^K$, then $(c(P), g) \in R' \forall g \in G_k(n)$. By Assumption 4 and the definition of $F_k(n)$,

$$\forall f \in F_k(n) \exists g \in G_k(n) \ni \quad \text{either } (1) f = g + h = g, \text{ or } (2) (g, f) \in R'.$$

In either case, $(c(P), f) \in R'$ by the transitivity of R'.

Since items are removed from *SOLUTION_COSTS* only if they are replaced by items that dominate them, the criterion for membership in *ND* implies that no node except those in the finite set T^K could be selected for expansion at any iteration greater than or equal to k. Since k is finite and only one node is selected for expansion per iteration, only a finite number of nodes could be expanded prior to iteration k. Therefore, if $N(P) \subseteq CLOSED_k$ for some iteration k, then the lemma is true.

For the second case, assume that no such k exists; that is, assume that $N(P) \nsubseteq CLOSED_k \forall k$. By Lemma 2(d), this implies that $N(P) \cap OPEN_k \neq \varnothing \forall k$. Let k be a specific such iteration and let n' be the shallowest node of P on $OPEN_k$. Let $g' \in G_k(n')$. By Assumption 3, there are numbers B and δ, such that

$$(g', c(P')) \in R' \forall P' \ni length\,\{P'\} \geq K_1,$$

where $K_1 = \mathit{length}\ \{P \cap \mathbf{P}(s, n')\}^* B/\delta$. By Assumptions 3 and 4, there is some finite positive integer, B', such that

$$\exists h \in H(n') \ni (h, c(P')) \in R' \forall P' \ni \mathit{length}\ \{P'\} \geq K_2,$$

where $K_2 = \mathit{length}\ \{P \cap \mathbf{P}(n', \Gamma)\}^* B'/\delta$. Let $h' \in H(n')$ be a particular such value, $K_3 = \mathit{length}\ \{P\}^*(B + B')/\delta \geq K_1 + K_2$, and let $n'' \in N(\mathbf{G}) \setminus T^{K_3}$. If no such n'' exists, the result is proved. Otherwise,

$$\forall f \in F_k(n'') \exists f' \in F_k(n') \ni (f', f) \in R',$$

as we now show.

By the definitions of $G_k(n'')$ and $F_k(n'')$, and by Assumption 4,

$$\forall f \in F_k(n'') \exists P'' \in \mathbf{P}(s, n'') \text{ and } g \in G_k(n'') \ni g = c(P'') \text{ and } (g, f) \in R.$$

Let $f' \in F_k(n'')$ and let $P'' \in \mathbf{P}(s, n'')$ be the corresponding path with cost $g'' = c(P'')$ so that $(g'', f') \in R$. Let P_1, P_2, and P_3 be proper subpaths of P'' such that $P'' = (P_1, P_2, P_3)$, $\mathit{length}\ \{P_1\} = K_1$, and $\mathit{length}\ \{P_2\} = K_2$; such subpaths exists by our selection of n''. Then we know

$$(c(P_1) + c(P_2), c(P'')) \in R'.$$

By Assumption 4, the order-preserving property of our cost and preference structure, and the expressions above, we also have

$$(g' + h', c(P_1) + h') \in R'$$

and

$$(c(P_1) + h', c(P_1) + c(P_2)) \in R'.$$

The transitivity of R' then implies that

$$(g' + h', c(P'')) \in R',$$

and

$$(g' + h', f'') \in R'.$$

By the definition of $F_k(n')$,

$$\forall g \in G_k(n') \text{ and } h \in H(n') \exists f \in F_k(n') \ni (f, g + h) \in R.$$

Let $f' \in F_k(n')$ be such that the last expression holds. Then transitivity again implies that

$$(f', f'') \in R'.$$

Since n'', f'', and k were chosen arbitrarily we have shown that

$$\forall n'' \in N(\mathbf{G}) \setminus T^{K_3} \text{ and } f' \in F_k(n'') \exists f \in F_k(n') \ni (f, f') \in R'.$$

Therefore, by the criterion by which nodes are selected for expansion, no node in the set $N(\mathbf{G}) \setminus T^{K_3}$ could ever be selected for expansion.

We have thus demonstrated that, if P is any solution path, then there exists a finite integer K, the size of which is dependent on the length of P, such that no node can ever be expanded that is not in T^K, which, by Assumption 1, is a finite set. □

We now present our first theorem, which states that MOA^* terminates on all graphs in which there exists a solution path.

THEOREM 1. *MOA^* terminates on any problem for which $\mathbf{P}(s, \Gamma) \neq \emptyset$.*

PROOF. The algorithm terminates when Step (2) is reached with $ND_k = \emptyset$. Since Step (2) is visited in each iteration of the algorithm, it is sufficient to show that $ND_k = \emptyset$ for some finite k. At each iteration in which the algorithm does not terminate, exactly one node is selected for expansion from ND. By Lemma 7, only a finite set of nodes are candidates for expansion. Therefore, if we show that each of these nodes may be selected for expansion only a finite number of times, then the result is proved.

The fact that any node may be selected for expansion only a finite number of times follows from the criterion for reopening a node and Lemma 3, since a new nondominated path from s to n must be discovered each time n is reopened and Lemma 3 implies that the number of nondominated paths from s to n is finite for any n. □

The next result provides the last ingredient needed for the proof of the completeness of MOA^*, by showing that, if P is any solution path, there will be a node of P on $OPEN$ at least until the first solution path is added to $SOLUTION$.

LEMMA 8. *If $\mathbf{P}(s, \Gamma) \neq \emptyset$, then*

$$SOLUTION_k = \emptyset \Rightarrow N(P) \cap OPEN'_k \neq \emptyset \forall k' \leq k \text{ and } P \in \mathbf{P}(s, \Gamma).$$

PROOF. According to the operation of the algorithm, if $SOLUTION_k = \emptyset$, then no goal node has been selected for expansion at any iteration $k' < k$. By the definition of node selection for expansion, this implies that

$$N(P) \cap \Gamma \notin CLOSED_{k'}, \forall k' \leq k \text{ and } P \in \mathbf{P}(s, \Gamma).$$

Therefore, by Lemma 2(d)

$$N(P) \cap OPEN_{k'} \neq \emptyset \forall k' \leq k \text{ and } P \in \mathbf{P}(s, \Gamma). \quad \square$$

We are now prepared to consider the completeness of the algorithm. According to the standard definition, an algorithm is said to be complete if it terminates with a solution whenever one exists. In multiobjective problems, we often seek a set of solutions, so the previous definition must be clarified as follows:

Definition. A search algorithm is *complete* if it produces at least one solution path as output whenever a solution path exists.

For problems such as those normally addressed by A^* in which a single solution path is sought, this definition of completeness reduces to the standard form found in [24, p. 75]. Note that neither the scalar nor the multiobjective forms of this definition require that the solution path found necessarily have any special characteristics; for example, it need not be a "good" solution path. Given the above preparation, the completeness of MOA^* is easily proven next.

THEOREM 2. *MOA^* is complete.*

PROOF. Assume $\mathbf{P}(s, \Gamma) \neq \emptyset$. In view of Theorem 1, it is sufficient to show that the algorithm cannot terminate without discovering a solution path;

that is, $SOLUTION \neq \varnothing$ at termination. By the termination condition, it is further sufficient to show that $ND = \varnothing \Rightarrow SOLUTION \neq \varnothing$ or, equivalently, that $SOLUTION = \varnothing \Rightarrow ND \neq \varnothing$. We can show this with two implications:

(a) $OPEN \neq \varnothing$ and $SOLUTION = \varnothing \Rightarrow ND \neq \varnothing$ and
(b) $SOLUTION = \varnothing \Rightarrow OPEN \neq \varnothing$.

Result (a) follows from the facts:

(i) When $SOLUTION = \varnothing$, ND is the set of nondominated nodes in $OPEN$.
(ii) If $OPEN \neq \varnothing$, then it contains at least one nondominated node, by Lemma 5. Result (b) is a direct consequence of Lemma 8. □

We now consider several results that characterize a traversal subgraph. The following lemma states that for every node n in the explicated subgraph, there will be at least one path from s to n in the traversal subgraph. Its corollary states the implication of this result in terms of the set of accrued path costs.

LEMMA 9. $\forall n \in ESG_k \exists P \in TSG_k \cap \mathbf{P}(s, n)$.

PROOF. As a base case for induction, note that for any n in $SCS(s)$, the path consisting of the single arc (s, n) is in $TSG_1 \cap \mathbf{P}(s, n)$. Therefore

$$\forall n \in ESG_1 \exists P \in TSG_1 \cap \mathbf{P}(s, n).$$

Assume at some later iteration k that

$$\forall n \in ESG_k \exists P \in TSG_k \cap \mathbf{P}(s, n).$$

We wish to show that

$$\forall n \in ESG_{k+1} \exists P' \in TSG_{k+1} \cap \mathbf{P}(s, n'). \tag{2}$$

If $n' \in ESG_k$, then by assumption $\exists P' \in TSG_k \cap \mathbf{P}(s, n')$. If, in addition, no parent of n' was expanded at iteration k, then the same path P' will remain in TSG_{k+1} and (2) will be true. Otherwise, if a parent of n' was expanded, but no backpointers from n' were moved, P' will again remain in TSG_{k+1} and (2) will be true. If a parent of n' was expanded at iteration k and backpointers from n' were moved, then either (1) $P' \in TSG_{k+1}$ or (2) $P' \notin TSG_{k+1}$, but

$$\exists P'' \in TSG_{k+1} \cap \mathbf{P}(s, n') \ni (c(P''), c(P')) \in R'.$$

Therefore, if $n' \in ESG_k$, then (2) holds.

If $n' \notin ESG_k$, then $\exists n'' \in ESG_k \ni n' \in SCS(n'')$ and n' was expanded at iteration k. By assumption, $\exists P'' \in TSG_k \cap \mathbf{P}(s, n'')$. Since $n' \notin ESG_k$, n' is a newly discovered node at iteration k, Step (5) (c) applies, and a backpointer is established from n' to n''. Therefore, by the definition of TSG_{k+1}, the path formed by adjoining P'' with (n'', n') will meet the appropriate conditions and once again, (2) will be true. Complete induction provides the desired result. □

COROLLARY 3. *If* $n \in ESG_k$, *then*

$$\exists g \in G_k(n) \text{ and } P \in TSG_k \cap \mathbf{P}(s, n) \ni g = c(P).$$

PROOF. Immediate from Lemma 9 and the definition of $G_k(n)$. □

The next result gives a set of conditions that are sufficient to guarantee that a particular path will be in a traversal subgraph.

LEMMA 10. *If $P \in \mathbf{P}^*(s, n)$, $N(P) \subseteq OPEN_K \cup CLOSED_k$, and $N(P) \cap OPEN_K \subseteq \{n\}$, then $P \in TSG_K$.*

PROOF. Let $P \in \mathbf{P}^*(s, n)$ and let n' be such that $(s, n') \in A(P)$. Note that $s \in CLOSED_1$, $n' \in OPEN_1$, and by Lemma 6, $(s, n') \in \mathbf{P}^*(s, n')$, so the construction of the algorithm ensures that $(s, n') \in TSG_1$.

At some later iteration k, let n'' be the deepest node of P to satisfy the assumptions of the lemma. The existence of at least one such node is guaranteed by the fact that n' will be such a node for all iterations after the first. Let $P' = P \cap \mathbf{P}(s, n'')$. Then $N(P') \subseteq OPEN_k \cup CLOSED_k$, $N(P') \cap OPEN_k \subseteq \{n''\}$, and assume that $P' \in TSG_k$. Note that under these assumptions, $P' \in TSG_{k'} \forall k' \geq k$, by Lemma 6 and the construction of the algorithm. Also note that $n'' \in OPEN_k$. Otherwise, $n'' \in CLOSED_k$ and, by Lemma 2(b), $N(P') \cup SCS(n'') \in OPEN_k \cup CLOSED_k$, violating our assumption that n'' was the deepest node of P to satisfy the conditions of the lemma.

If $n'' = n$, then the result is proved. Otherwise, since $n'' \in CLOSED_K$, by the definition of $CLOSED_K$, there is at least one iteration after k and before K in which n'' is expanded. Let k'' be the first such iteration. This implies that $n'' \in CLOSED_{k''+1}$. If P'' is rediscovered at iteration k', then $P'' \in TSG_k''$. Otherwise, in Step (5(c)) a backpointer is established from n''' to n'', n''' is added to $OPEN_{k''+1}$, and $P'' \in TSG_{k''+1}$. The result follows by complete induction. □

Note that the construction of the algorithm and the definitions of TSG_k and $G_k(n)$ imply the following corollary to Lemma 10.

COROLLARY 4. *If $P \in \mathbf{P}^*(s, n)$, $N(P) \subseteq OPEN_k \cup CLOSED_k$, and $N(P) \cap OPEN_k \subseteq \{n\}$, then $c(P) \in G_k(n)$.*

Our next result states that a nondominated solution path will have at least one node on *OPEN* in every iteration before it is placed on *SOLUTION*. Note its similarity with Lemma 8.

LEMMA 11. *If $P \in \mathbf{P}^*(s, \Gamma)$ and $P \notin SOLUTION_k$, then $N(P) \cap OPEN_k \neq \varnothing$.*

PROOF. By Lemma 2(d), either $N(P) \subseteq CLOSED_k$, or $N(P) \cap OPEN_k \neq \varnothing$, so it is sufficient to show that

$$P \notin SOLUTION_k \Rightarrow N(P) \nsubseteq CLOSED_k.$$

Equivalently, we may show that

$$N(P) \subseteq CLOSED_k \Rightarrow P \in SOLUTION_k.$$

To show this last implication, note that if $N(P) \subseteq CLOSED_k$, Lemma 10 applies, so that $P \in TSG_k$. Let n be the deepest nongoal node of P. The only situation in which P could have been added to TSG is if $\gamma = N(P) \cap \Gamma$ were selected for expansion while $P' \in TSG$, where $P' P \cap \mathbf{P}(s, n)$. In this case, Step (4) of the algorithm would apply and P would be added to *SOLUTION*. □

The following lemma states that if n' is the shallowest open node on some nondominated path, then the algorithm has already discovered a nondominated path to n' and that this path will remain nondominated in $\mathbf{P}(s, n')$ throughout the remainder of the search.

LEMMA 12. *Let n' be the shallowest node in $OPEN_k$ on a nondominated path $P \in \mathbf{P}^*(s, n)$ and let $P' = \mathbf{P} \cap \mathbf{P}(s, n')$. Then $c(P') \in G^*(n)$ and*

$$c(P') \in G_{k'}(n') \forall k' \leq k.$$

PROOF. By Lemma 6, Lemma 10, and Corollary 4, $P' \in TSG_k$ and $c(P') \in G_k(n')$, and $P' \in \mathbf{P}^*(s, n')$. The result follows from these facts, the definitions of g and g^*, and the fact that paths in $\mathbf{P}^*(s, n')$ can never be removed from TSG. □

We conclude this section with two simple, but useful, facts. The first one follows directly from our definition of ND. It is useful in the proof of Theorem 3 below.

Fact 3. $ND_k = \emptyset$ if and only if the following condition holds:

$$\forall f \in F_k \exists c^* \in SOLUTION\_COSTS_k \ni (c^*, f) \in R'.$$

The following fact is a simple consequence of our definitions of *SOLUTION_COSTS*, $\mathbf{P}^*(s, \Gamma)$, and our dominance relation.

Fact 4. For all iterations k,

$$\forall c' \in SOLUTION\_COSTS_k \exists P \in \mathbf{P}^*(s, \Gamma) \ni (c(P), c') \in R.$$

The above results concerning MOA^* are "independent" of the characteristics of the set of heuristic functions used; that is, the only required restriction on the heuristic functions is Assumption 4. Most of the results in the remainder of this paper presume that the set of heuristics used with MOA^* is admissible, a property considered in the next section.

4.3. ADMISSIBILITY.

One of the most useful properties of A^* is admissibility. Loosely speaking, admissibility means that A^* is guaranteed to discover a desired solution for any single-objective path problem. The following definition extends this concept to the general case of problems to which MOA^* is applicable.

Definition. An *admissible algorithm* is an algorithm that is guaranteed to terminate with the entire set of nondominated solution paths on any problem in which at least one solution path exists.

Using the following definition of an admissible set of heuristics, we show that MOA^* has this property.

Definition. An *admissible set of heuristic functions*, (with respect to the set of nondominated costs $\{H^*(n): n \in N(\mathbf{G})\}$) is a set of heuristic functions that meets the following condition:

$$\forall P \in \mathbf{P}^*(s, \Gamma),\ n \in N(P), \text{ and } h^* \in H^*(n) \exists h \in H(n) \ni (h, h^*) \in R. \quad (3)$$

An obviously sufficient condition for admissibility that may be somewhat easier to verify than our definition is that (3) holds for all $n \in N(\mathbf{G})$. Based on this definition, the following assumption is made for the remainder of the results in this section, unless otherwise noted:

Assumption 8. The set of heuristics used in MOA^* is admissible.

The next lemma states that at any iteration before the algorithm terminates, very nondominated solution path that has not been added to *SOLUTION* has a node on *OPEN* with a node selection value that is not dominated by the actual cost of that solution path.

LEMMA 13. *For all iterations k, if $P \in \mathbf{P}^*(s, \Gamma) \setminus SOLUTION_k$, then*

$$\exists n \in N(P) \cap OPEN_k \text{ and } f \in F_k(n) \ni (f, c(P)) \in R.$$

PROOF. Consider any $P \in \mathbf{P}^*(s, \Gamma)$ at any k for which $P \notin SOLUTION_k$. Lemma 12 ensures that $N(P) \cap OPEN_k \neq \emptyset$. The result holds trivially if $k = 0$; assume $k > 0$. Let n' be the shallowest node of P on $OPEN_k$ and let $P' = P \cap \mathbf{P}(s, n')$.

By assumption, P is a nondominated solution path so that, according to the definition of C^*, $c(P) \in C^*$. By Lemma 13, we may select $g' \in G_k(n')$ and $g^* \in G^*(n')$ such that

$$g' = c(P') = g^*.$$

Now let $P'' = P \cap \mathbf{P}^*(n', \Gamma)$, so that $P = (P', P'')$. By the definition of $H^*(n')$,

$$\exists h^* \in H^*(n') \ni h^* = c(P''). \tag{4}$$

Since H is admissible, the h^* satisfying (4) is such that

$$\exists h \in H(n') \ni (h, h^*) \in R. \tag{5}$$

Let $h' \in H(n')$ be chosen so that (5) is satisfied; that is, so that $(h, h^*) \in R$. Combining with the earlier expression for g' using the order-preserving nature of our cost and preference structure, we get

$$(g' + h', c(P)) \in R.$$

By the definition of $F_k(n')$,

$$\exists f' \in F_k(n') \ni (f', g' + h') \in R. \tag{6}$$

Combining this last expression with (6), and using the transitivity of the relation R, we get

$$\exists f' \in F_k(n') \ni (f', c(P)) \in R,$$

which is the desired result. □

We now present the major result of this paper.

THEOREM 3. *MOA^* is an admissible algorithm whenever it operates with an admissible set of heuristics.*

PROOF. Theorem 1 indicates that it is sufficient to prove that the algorithm will not terminate until all nondominated solution paths and no others have been discovered and placed in *SOLUTION*; that is, until $\mathbf{P}^*(s, \Gamma) = SOLUTION$.

Therefore, Fact 4 implies that it is also sufficient to show that, if the algorithm terminates in iteration K, then $\mathbf{P}^*(s, \Gamma) \subseteq SOLUTION_K$. By the termination condition, we need only show that, if there is a member of $\mathbf{P}^*(s, \Gamma)$ that is not yet in $SOLUTION$, then ND will not be empty. By Lemma 12

$$\exists P \in \mathbf{P}^*(s, \Gamma) \setminus SOLUTION_k$$
$$\Rightarrow \exists n \in N(P) \cap OPEN_k \text{ and } f \in F_k(n) \ni (f, c(P)) \in R.$$

By the definitions of C^* and $\mathbf{P}^*(s, \Gamma)$.

$$c(P) \in C^* \Leftrightarrow \nexists P' \in \mathbf{P}(s, \Gamma) \ni (c(P'), c(P)) \in R'.$$

Since $SOLUTION_k \subseteq \mathbf{P}(s, \Gamma)$ by definition, we therefore have

$$\nexists P' \in SOLUTION_k \ni (c(P'), f) \in R'.$$

Since $SOLUTION\_COSTS_k \cup \{f\} \subseteq C_k$, transitivity implies that

$$\nexists P' \in SOLUTION_k \ni (c(P'), f) \in R'.$$

Thus n and its associated f do not allow the conditions in Fact 3 to be met and hence, $ND_k \neq \emptyset$. □

4.4. Node Expansion and Comparison of Heuristics. According to the above definitions of admissibility, a set of heuristic functions, H, consisting solely of a zero function ($H = \{h\colon\ h_i = 0 \forall n \in N$ and $i \in \{1, \ldots, M\}\}$) is admissible. Obviously, this will not yield a very informed search. This observation leads us to investigate some formal means of comparing sets of heuristic functions as well as the versions of the algorithm that they direct. In this section, we compare admissible heuristics and algorithms according to the number of nodes they expand, paralleling the development of similar results for A^*.

The significance of this comparative measure varies widely among different problem domains. Ultimately, it may be of greater interest to draw such comparisons based on the more common fundamental measures of computational complexity in computer science; that is, time and space complexity.

We begin with several definitions.

Definition. A *set of heuristic functions H* is said to be *at least as informed as* heuristic function set H' if both are admissible and for all $n \in N(\mathbf{G})$ the following condition holds:

$$\forall h' \in H'(n), \exists h \in H(n) \ni (h, h') \in R.$$

Definition. Search algorithm A using H is said to be *at least as informed as* the version of that algorithm A' using H' when H is at least as informed as H'.

Definition. Search algorithm A is said to be *nondominated* with respect to search algorithm A' if, in every problem to which both A and A' apply, every node selected for expansion by A before termination must also be selected for expansion by A' before termination.

Since the number of nodes expanded is our comparative measure, we first consider several results associated with node expansion. Our next lemma gives a necessary condition for a node to be selected for expansion. Note that this condition depends upon the stage of the search; that is, on the iteration k.

LEMMA 14. *If MOA* selects a node n for expansion at iteration k, then*

$$\exists f \in F_k(n) \ni \nexists P \in \mathbf{P}(s, \Gamma) \ni (c(P), f) \in R'.$$

PROOF. Assume to the contrary that MOA^* operates with an admissible set of heuristics and selects n for expansion at iteration k, but

$$\forall f \in F_k(n) \exists P \in \mathbf{P}(s, \Gamma) \ni (c(P), f) \in R'.$$

By the definition of $\mathbf{P}^*(s, \Gamma)$, this implies that

$$\forall f \in F_k(n) \exists P \in \mathbf{P}^*(s, \Gamma) \ni (c(P), f) \in R'. \tag{7}$$

For some $f \in F_k(n)$, let P' satisfy (7). If $P' \in SOLUTION_k$, f cannot cause n to be added to ND_k due to the definition of ND_k. If $P' \notin SOLUTION_k$, then Lemma 12 applies and

$$\exists n' \in N(P') \cap OPEN_k \text{ and } f' \in F_k(n') \ni (f', c(P')) \in R.$$

Together with (7), this implies that $(f', f) \in R'$. Thus, the existence of n' on $OPEN_k$ means that f cannot cause n to be added to ND_k, again by the definition of ND_k. Therefore, since f was selected arbitrarily, no $f \in F_k(n)$ can cause n to be added to ND_k. According to the definition of node selection for expansion, this contradicts our assumption that n was selected for expansion at iteration k, thus proving the desired result. □

The following lemma provides the basis for its successor, which defines a sufficient condition for a node to be selected for expansion.

LEMMA 15. *Assume MOA* terminates at iteration k. Then*

$$\forall n \in OPEN_k \nexists (f, P) \in F_k(n) \times \mathbf{P}^*(s, \Gamma) \ni (f, c(P)) \in R.$$

PROOF. Assume

$$\exists n \in OPEN_k, f \in F_k(n), \text{ and } P \in \mathbf{P}^*(s, \Gamma) \ni (f, c(P)) \in R.$$

By the definition of $\mathbf{P}^*(s, \Gamma)$, this implies that

$$\nexists P' \in \mathbf{P}^*(s, \Gamma) \ni (c(P'), f) \in R'.$$

By Fact 4, we therefore have

$$\nexists P' \in SOLUTION_k \ni (c(P'), f) \in R'.$$

Therefore, by Fact 3, $ND_k \neq \emptyset$ and the termination condition cannot be met. □

Note that, similar to the necessary condition in Lemma 14, the following sufficient condition for a node to be selected for expansion depends on the stage of the search; that is, on the iteration k.

LEMMA 16. *Assume*

$$n \in OPEN_k \text{ and } \exists f \in F_k(n) \text{ and } P \in P^*(s, \Gamma) \ni (f, c(P)) \in R.$$

Then n will be selected for expansion before termination.

PROOF. Assume

$$n \in OPEN_k \text{ and } \exists f \in F_k(n) \text{ and } P \in P^*(s, \Gamma) \ni (f, c(P)) \in R.$$

By Lemma 15, n will be removed from $OPEN$ before termination. Therefore, since the algorithm ensures that nodes leave $OPEN$ only by being selected for expansion, n will be selected for expansion before termination. □

We now develop a necessary and sufficient condition for a node to be selected for expansion that is independent of the iteration. The condition is based on the following definition, which is a generalization of the term C-bounded as defined by [24, p. 80].

Definition. Assume the following are given:

—a finite set $X \subseteq \Re^{M++}$
—an order relation R and its induced relation R' defined on X, and
—a finite set of heuristic functions H.

A path P, originating at the start node, is said to be *X-nondominated* (with respect to H) if

$$\forall n \in N(P) \exists P' \in \mathbf{P}(s, n) \text{ and } h \in H(n) \ni \nexists x \in X \ni (x, c(P') + h) \in R'.$$

The next result provides a necessary and sufficient condition for MOA^* to select a node for expansion.

LEMMA 17. *In MOA^*, a node n will be selected for expansion before termination if and only if $\exists P \in \mathbf{P}(s, n)$ such that P is C^*-nondominated with respect to H.*

PROOF. Assume there exists a C^*-nondominated path $P \in \mathbf{P}(s, n)$, but the algorithm terminates at iteration k and n has not been selected for expansion. If the algorithm terminates at iteration k, then $ND_k = \varnothing$. If n has not been selected for expansion before termination, then $n \notin CLOSED_k$. Therefore, by Lemma 1(d), $N(P) \cap OPEN_k \neq \varnothing$. Let n' be the shallowest open node of P. By Fact 3

$$\forall f \in F_k(n') \exists P' \in SOLUTION_k \ni (c(P'), f) \in R'.$$

Using this along with Fact 4, we have

$$\forall f \in F_k(n') \exists c \in C^* \ni (c, f) \in R'.$$

By the definition of $F_k(n')$,

$$\forall g \in G_k(n') \text{ and } h \in H(n') \exists f \in F_k(n') \ni (f, g + h) \in R,$$

and conversely

$$\forall f \in F_k(n') \exists g \in G_k(n') \text{ and } h \in H(n') \ni f = g + h.$$

Thus,

$$\forall g \in G(n') \text{ and } h \in H(n') \exists c \in C^* \ni (c, g + h) \in R'.$$

Let $P'' = P \cap \mathbf{P}(s, n')$. By Lemma 9, since n' is the shallowest node of P on $OPEN_k$, P'' has been discovered and therefore,

$$\exists g' \in G_k(n') \ni (g', c(P'')) \in R'.$$

By the order-preserving nature of our cost and preference structure, this implies that

$$\exists g' \in G_k(n') \ni (g' + h, c(P'') + h) \in R' \forall h \in H(n').$$

In either case, the transitivity of R and R' implies that

$$\forall h \in H(n') \exists c \in C^* \ni (c, c(P'') + h) \in R'.$$

This contradicts our assumption that P was C^*-nondominated, thus proving the sufficiency of the condition.

To prove the necessity of the stated condition, assume that the algorithm selects node n for expansion at iteration k and let $PP \in \mathbf{PP}_k(n, s)$; existence is ensured by Lemma 7. By the construction of the algorithm, each node $n' \in N(PP) \setminus \{n\}$ was expanded in some iteration prior to k. Let $n' \in N(PP) \setminus \{n\}$ and let k' be the iteration in which it was expanded. By Lemma 14,

$$\exists f \in F_{k'}(n') \ni \nexists P \in \mathbf{P}(s, \Gamma) \ni (c(P), f) \in R'. \tag{8}$$

Let $f' \in F'_k(n')$. By the definition of $F'_k(n')$,

$$\exists g' \in G'_k(n') \text{ and } h' \in H(n') \ni f' = g' + h'.$$

By the construction of the algorithm, this implies

$$\exists g'' \in G_k(n') \ni (g'', g') \in R.$$

Therefore, by the order preserving nature of our cost and preference structure and the definition of $F_k(n')$,

$$\begin{aligned}\exists f'' \in F_k(n') \ni \text{ either } &(1)\ f'' = f' = g' + h', \\ \text{or } &(2)\ (f'', f') \in R' \text{ and } f'' = g'' + h'.\end{aligned}$$

Together with (8), using transitivity for case (2), this implies that

$$\exists f'' \in F_k(n') \ni \exists P \in \mathbf{P}(s, \Gamma) \ni (c(P), f'') \in R',$$

or, by the definition of g'',

$$\exists P' \in \mathbf{P}(s, n') \text{ and } h \in H(n') \ni \nexists P \in \mathbf{P}(s, \Gamma) \ni (c(P), c(P') + h) \in R'.$$

By the definition of C^*, this last statement provides the desired result. □

The following theorem provides a means of comparing the performance of versions of MOA^* that use different sets of heuristic functions, provided that the sets of heuristic functions are comparable in terms of the definition of "informed" above. Thus, a partial order can be defined on the space of heuristic sets or, equivalently, on the space of algorithms using those heuristic sets, in terms of the notion of "informed."

THEOREM 4. *Let A and A' be two versions of MOA^* differing only in their heuristic function sets. If A is at least as informed as A', then A is nondominated with respect to A'.*

PROOF. Let algorithm version A use admissible heuristic set H and A' use H'. Assume that H is at least as informed as H' and A expands node n. By Lemma 14, there exists $P \in \mathbf{P}(s, n)$ such that P is C^*-nondominated with respect to H. Since H is at least as informed as H, P will also be C^*-nondominated with respect to H'. Therefore, by Lemma 14, n will also be expanded by A'. □

4.5. MONOTONE AND CONSISTENT HEURISTICS. In single objective search using A^*, the use of heuristic functions with certain special properties provides significant advantages in that such functions allow the algorithm to be greatly simplified. This simplification is due to the fact that, when it is used with a heuristic function having these special characteristics, A^* will never reopen a node. Unfortunately, the natural extensions of these special properties to our general case do not provide as much advantage because they are not sufficient to guarantee that nodes will not need to be reopened. In this section, we analyze the implications that the use of heuristics possessing these special properties, monotonicity and consistency, has in the general case. We begin by providing generalized definitions of "monotone" and "consistent" sets of heuristics.

Definition. A set of heuristic functions H is said to be *monotone* with respect to the associated cost and preference structure if, for all nodes $n \in N(\mathbf{G}) \setminus \{s\}$, the following condition holds:

$$\forall h \in H(n) \text{ and } n' \in SCS^{-1}(n) \exists h' \in H(n') \ni (h', c(n', n) + h) \in R.$$

Definition. A set of heuristic functions is said to be *consistent* with respect to the associated cost and preference structure if, for all nodes $n \in N(\mathbf{G}) \setminus \{s\}$, the following condition holds:

$$\forall h \in H(n),\ n' \in \{\text{ancestors of } n\}, \text{ and}$$
$$P \in \mathbf{P}^*(n', n) \exists h' \in H(n') \ni (h', c(P) + h) \in R.$$

The equivalence of monotonicity and consistency for sets of heuristic functions is the subject of the next lemma. The scalar equivalent of this result was recently rediscovered by [18], having been presented previously by [24].

LEMMA 18. *A set of heuristic functions is consistent if and only if it is monotone.*

PROOF. In the definition of a consistent set of heuristic functions, select n' to be a parent of n. If the path represented by arc (n, n') is in $\mathbf{P}^*(n, n')$, then the monotonicity condition is met. Otherwise, by the definition of $\mathbf{P}^*(n, n')$,

$$\exists P' \in \mathbf{P}^*(n, n') \ni (c(P'), c(n, n')) \in R',$$

so, by the order-preserving nature of the cost and preference structure, the monotonicity condition again follows directly.

To prove the converse, assume H is monotone with respect to the current cost and preference structure. Choose any $n \neq s$ and let $n' \in N(\mathbf{G})$ be any ancestor of n. Choose any $P \in \mathbf{P}^*(n', n)$ and let K be such that $P = (n_0, n_1, \ldots, n_{K-1})$, where $n_0 = n'$ and $n_{K-1} = n$. By monotonicity, there exist $h(n_i) \in H(n_i)$, $i = 0, 1$, such that

$$(h(n_0), c(n_0, n_1) + h(n_1)) \in R,$$

so the result holds when $K = 2$.

Assume $(h(n_0), c(P') + h(n_{k-1})) \in R$, where $P' = P \cap \mathbf{P}(n_0, n_{k-1})$. By monotonicity, there is an $h(n_k) \in H(n_k)$ such that,

$$(h(n_{k-1}), c(n_{k-1}, n_k) + h(n_k)) \in R.$$

Note that

$$c(P'') = c(P') + c(n_{k-1}, n_k),$$

where $P'' = P \cap \mathbf{P}(n_0, n_k)$, so

$$(h(n_0), c(P'') + h(n_k)) \in R$$

and the set of heuristics is consistent. The result follows by complete induction. □

The equivalent properties of monotonicity and consistency of heuristic sets also imply the other significant characteristic that a set of heuristic functions may possess, admissibility, as is seen in the next lemma.

LEMMA 19. *A monotone set of heuristics is also admissible.*

PROOF. In view of Lemma 18, it is sufficient to show that a consistent set of heuristic functions is also admissible. In the definition of a consistent set of heuristic functions, let $n \in \Gamma$. Then $h \in H(n)$ is defined by $h_i = 0 \forall i \in \{1, \ldots, M\}$ and the definition of a consistent set of heuristic functions yields

$$\forall n' \in \{\text{ancestors of } n\} \text{ and } P \in \mathbf{P}^*(n', \Gamma) \exists h' \in H(n') \ni (h', c(P)) \in R.$$

For admissibility, we must show

$$\forall n \in \{\cup N(P)\colon P \in \mathbf{P}^*(s, \Gamma)\} \text{ and } h^* \in H^*(n) \exists h \in H(n) \ni (h, h^*) \in R.$$

Note that,

$$\{\cup N(P)\colon P \in \mathbf{P}^*(s, \Gamma)\} \subset \{\cup \{\text{ancestors of } n\}\colon n \in \Gamma\}$$

and the definition of $H^*(n')$ states that

$$H^*(n') = \{c(P)\colon P \in \mathbf{P}^*(n', \Gamma)\},$$

so that the definition of a consistent set of heuristic functions used on each node in Γ provides the desired result. □

Based on Lemma 19, we now replace Assumption 8 with the following assumption for the remainder of the results in this paper, unless indicated otherwise.

Assumption 9. The set of heuristics used in MOA^* is monotone.

The following lemma asserts that a version of the algorithm guided by a monotone set of heuristics has discovered at least one nondominated path to each node it expands.

LEMMA 20. *If a node n is selected for expansion in iteration k, then*

$$G_k(n) \cap G^*(n) \neq \varnothing.$$

PROOF. Assume that $n \in OPEN_k$ is selected for expansion at some iteration k, and that $G_k(n) \cap G^*(n) = \emptyset$. By the definition of $G^*(n)$, this implies

$$\forall g \in G_k(n) \exists P \in \mathbf{P}^*(s, n) \text{ and } g^* \in G^*(n) \ni c(P) = g^* \text{ and } (g^*, g) \in R'. \tag{9}$$

Let $g \in G_k(n)$ and let $P \in \mathbf{P}^*(s, n)$ be such that (9) holds. Note that $n \in N(P) \cap OPEN_k$. If $N(P) \cap OPEN_k = \{n\}$, then by Lemma 8,

$$G_k(n) \cap G^*(n) \neq \emptyset,$$

which is a contradiction, so the result holds.

Now assume $N(P) \cap OPEN_k \neq \{n\}$ and let n' be the shallowest node in $N(P) \cap OPEN_k$. We now show that

$$\forall f \in F_k(n) \exists f' \in F_k(n') \ni (f', f) \in R'.$$

By Fact 2 and the definition of ND_k, this last equation contradicts the assumption that n was selected for expansion at iteration k. By Lemma 8,

$$\exists g' \in G_k(n') \ni g' = c(P \cap \mathbf{P}(s, n')) \in G^*(n').$$

Thus, using our assumed additive cost structure, we have

$$\begin{aligned} c(P) &= c(P \cap \mathbf{P}(s, n')) + c(P \cap \mathbf{P}(n', n)) \\ &= g' + c(P \cap \mathbf{P}(n', n)). \end{aligned}$$

By Lemmas 2 and 18 and the definition of a consistent set of heuristic functions,

$$\forall h \in H(n) \exists h' \in H(n') \ni (h', c(P \cap \mathbf{P}(n', n)) + h) \in R.$$

Combining the last two expressions and rearranging terms, we get

$$\forall h \in H(n) \exists h' \in H(n') \ni (g' + h', c(P) + h) \in R.$$

Note that (9) implies, using order-preservation, that

$$(g^* + h, g + h) \in R',$$

and since $c(P) + h = g^* + h$, by transitivity we have

$$(g' + h', g + h) \in R'.$$

By the definition of $F_k(n')$,

$$\exists f' \in F_k(n') \ni (f', g' + h') \in R.$$

Therefore, by transitivity,

$$\exists f' \in F_k(n') \ni (f', g + h) \in R'.$$

Since g and h were selected arbitrarily and

$$\forall f \in F_k(n) \exists g \in G_k(n) \text{ and } h \in H(n) \ni f = g + h,$$

the result is proved. □

In the case of A^*, Lemma 20 implies that nodes are never reopened if a monotone heuristic function is used, allowing the algorithm to be considerably simplified; that is, the significant work of Step (5.3.2) can be eliminated, as well as the test in Step (5.3.1). This leads us to investigate the possibility of similar simplifications in MOA^* when monotone heuristics are available.

Unfortunately, it may be necessary to reopen and/or reexpand nodes in a search involving multiple objectives, even when monotone heuristics are used. Thus, the situation is considerably complicated by the potential for multiple nondominated paths and path costs. This complication manifests itself throughout the work on the multiobjective case, including the next two theorems. These theorems are presented without proof, since the proofs are long and complex and they add little to the development. See [26] for the associated proofs.

In the scalar case, it can be shown that the sequence of node selection values generated during a search directed by a monotone set of heuristics will be nondecreasing in iteration count. The following two theorems give the generalization of this result in the current situation. Together, the next two theorems essentially state that the algorithm will not select any node for expansion to which a node selected for expansion at a later iteration is strictly preferred.

THEOREM 5. *Assume that* $\{n_0, n_1, n_2, \ldots\}$ *is a sequence of nodes selected for expansion in iterations* $\{0, 1, 2, \ldots\}$ *of the search. Then for all nonnegative integers i there is some* $f \in F_i(n_i)$ *such that there will not exist any* $f' \in F_{i+1}(n_{i+1})$ *for which* $(f', f) \in R'$.

The search problem shown in Figure 5 and Tables III and IV provides a counterexample to the claim that the statement of Theorem 5 holds, in general, if $i + 1$ is everywhere replaced by $j > i$. The simple two-objective shortest-path problem of this example thus demonstrates the necessity of the condition in the statement of Theorem 6 below that n_j not be a reexpanded node. It also illustrates the fact that nodes may, in general, need to be reexpanded during a multiobjective search with monotone heuristics.

The monotone set of heuristics used for this problem was constructed strategically to demonstrate the desired behavior; that is, reopening of nodes, etc. The solutions to Example 2 are as follows:

$$\begin{aligned}
P_1^* &= (s, 1, 4, \gamma) & c(P1^*) &= (6\ \ 10)\\
P_2^* &= (s, 2, 1, 4, \gamma) & c(P2^*) &= (9\ \ 9)\\
P_3^* &= (s, 3, 5, \gamma) & c(P3^*) &= (6\ \ 10).
\end{aligned}$$

Theorem 6 gives the required extension of Theorem 5 as indicated by Example 2.

THEOREM 6. *Assume that* $\{n_0, n_1, n_2, \ldots\}$ *is a sequence of nodes selected for expansion in iterations* $\{0, 1, 2, \ldots\}$ *of the search. Then the following condition holds for all nonnegative integers i and j such that* $i < j$ *and such that* $n_j \neq n_k \forall k < i$ (*this last condition ensures that* n_j *has not been previously selected for expansion*):

$$\exists f \in F_i(n_i) \ni \exists f' \in F_j(n_j) \ni (f', f) \in R'.$$

Although these last two theorems do not provide sufficient structure to allow a simple modification of MOA^* along the lines of what can be done in the case

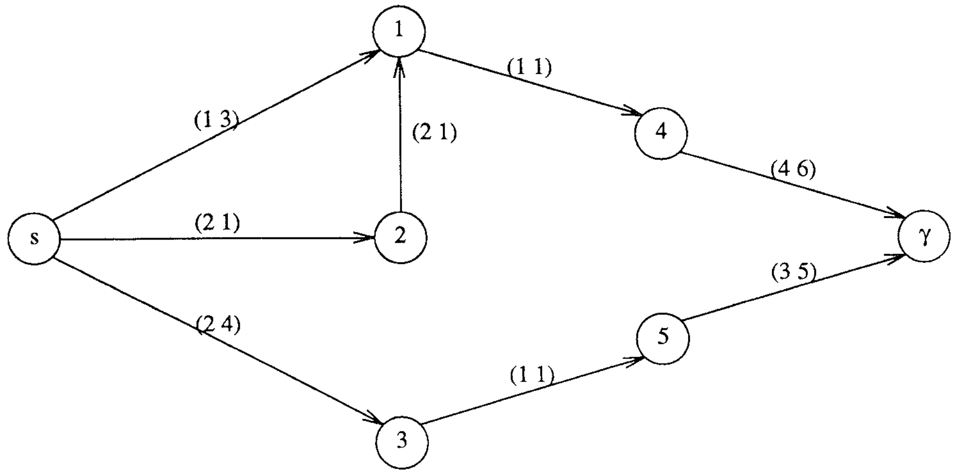


FIG 5 State-space graph for example 3

TABLE III. DERIVED COSTS AND HEURISTIC FUNCTION VALUES FOR EXAMPLE 2.

	Derived Costs			Heuristic Values
Node	$G^*(n)$	$H^*(n)$	$F^*(n)$	$H(n)$
s	(0 0)	(6 10) (9 9)	$H^*(s)$	(3 4)
1	(1 3) (4 2)	(5 7)	(6 10) (9 9)	(2 3)
2	(2 1)	(7 8)	(9 9)	(4 4)
3	(2 4)	(4 6)	(6 10)	(2 2)
4	(2 4)	(4 6)	(6 10)	(3 5)
	(5 3)		(9 9)	
5	(3 5)	(3 5)	(6 10)	(3 5)
γ	(6 10) (9 9)	(0 0)	(6 10) (9 9)	(0 0)

of A^*, there is still some possibility that they will be useful in identifying more complex computational enhancements to MOA^* when monotone heuristics are available.

Our last theorem contains the most useful necessary and sufficient condition for selecting a node for expansion when the algorithm is used with a monotone set of heuristics. Note that the necessity of the stated condition does not require the monotonicity assumption.

THEOREM 7

(*a*) *Assume that MOA^* operates with an admissible set of heuristics H (Assumption 8) and selects a node n for expansion. Then,*

$$\exists g \in G^*(n) \text{ and } h \in H(n) \ni \exists c \in C^* \ni (c, g + h) \in R'. \qquad (10)$$

TABLE IV. Numerical Trace of the Solution to the Example 2.

	$OPEN_k$			Backpointer Adjustments			
k	n	New $G_k(n)$	New $F_k(n)$	n'	$LABEL_k(n', n)$	$CLOSED_k$	$SOLUTION\_COSTS_k$ and $GOALS_k$
0	s^{**}	(0 0)	(3 4)			$\varnothing$	$\varnothing$
1	1^{**}	(1 3)	(3 6)	s	(1 3)	s	$\varnothing$
	2^{*}	(2 1)	(6 5)	s	(2 1)		
	3	(2 4)	(4 6)	s	(2 4)		
2	$2^{*}, 3^{**}$					s, 1	$\varnothing$
	4	(2 4)	(5 9)	1	(2 4)		
3	$2^{**}, 4^{*}$					s, 1, 3	$\varnothing$
	5	(3 5)	(6 10)	3	(3 5)		
4	1^{**}	(4 2)	(6 5)	2	(4 2)	s, 2, 3	$\varnothing$
	4, 5						
5	4^{**}	(5 3)	(8 8)	1	(2 4)(5 3)	s, 1, 2, 3	$\varnothing$
	5						
6	5^{*}					s, 1, 2, 3, 4	$\varnothing$
	γ^{**}	(6 10)(9 9)	(6 10)(9 9)	4	(6 10)(9 9)		
7	5^{**}					s, 1, 2, 3 4, γ	$\{(6\ 10), (9\ 9)\}$; $\{\gamma\}$
8	γ^{**}			5	(6 10)	2, 1, 2, 3 4, 5	$\{(6\ 10), (9\ 9)\}$, $\{\gamma\}$
9	$\varnothing$					s, 1, 2, 3, 4, 5, γ	$\{(6\ 10), (9\ 9)\}$; $\{\gamma\}$

(*b*) *Assume MOA* operates with a monotone set of heuristics H* (*Assumption* 9) *and eq.* (10) *holds for some node n. Then, MOA* will select n for expansion.*

Proof

(a) By Lemma 11, if n is selected for expansion, then there is some iteration k for which

$$\exists f \in F_k(n) \ni \exists c \in C^* \ni (c, f) \in R'. \tag{11}$$

By the definition of G^*,

$$\forall g \in G_k(n) \exists g^* \in G^*(n) \ni (g^*, g) \in R.$$

Therefore, by the definition of $F_k(n)$ and order-preservation

$$\forall f \in F_k(n) \exists g^* \in G^*(n) \text{ and } h \in H(n) \ni (g^* + h, f) \in R. \tag{12}$$

The result is now derived using transitivity and (11) as follows. Assume to the contrary that

$$\exists c \in C^* \ni (c, g^* + h) \in R'.$$

Then transitivity along with (12) yields $(c, f) \in R'$, contradicting (11) and proving the desired result.

(b) Assume H is monotone. For any $n \in N(\mathbf{G})$ such that $\mathbf{P}(s, n) \neq \emptyset$, let $P \in \mathbf{P}^*(s, n)$, with existence of such a P ensured by Lemma 1. By Lemma 8 and the definition of G^*,

$$\forall n' \in N(P) \exists g \in G^*(n)$$

and

$$g' \in G^*(n') \ni g = c\big(P \cap \mathbf{P}(n', n)\big) + g'.$$

By Lemma 18 and the definition of a consistent set of heuristic functions,

$$\forall h \in H(n) \exists h' \in H(n') \ni \big(h', h + c\big(P \cap \mathbf{P}(n', n)\big)\big) \in R.$$

Combining these last two expressions, using transitivity and order-preservation where necessary, yields

$$\forall n' \in N(P) \exists g \in G^*(n),\ g' \in G^*(n'),\ h \in H(n),$$

and

$$h' \in H(n') \ni g + h = g' + c\big(P \cap \mathbf{P}(n', n)\big) + h,$$

and

$$(g' + h', g + h) \in R.$$

So, if

$$\exists g \in G^*(n) \qquad \text{and} \qquad h \in H(n) \ni \exists c \in C^* \ni (c, g + h) \in R',$$

then a similar condition holds for all ancestors of n along the corresponding path $P \in \mathbf{P}^*(s, n)$ with $g = c(P)$; existence of such a corresponding P is ensured by the definition of $G^*(n)$. Therefore, such a P will be C^*-nondominated and, by Lemma 14, n will be selected for expansion. □

5. *Conclusions*

This paper has presented initial results in a study of multiobjective search. We defined an abstract multiobjective search problem and outlined an adaptation of the single-objective A^* algorithm, MOA^*, for solving it. The behavior of the algorithm was briefly illustrated on a simple two-objective shortest path problem. By suitably adapting the terminology used in work on A^*, we showed that the valuable properties of completeness and admissibility are inherited by MOA^*. We then proved that sets of heuristics, and the versions of MOA^* that used them, can be compared based on the number of nodes selected for expansion. Finally, we found that, while the properties of monotonicity and consistency are not as important as they are in single objective search, these properties still have some implications of interest in the multiobjective case.

There are two basic directions for future research in the area of multiobjective search. On the theoretical side, much additional work must be done to complete the development of MOA^*. Under some specific assumptions about the information available to aid in the search process, it can be shown that A^* is optimal among several classes of search algorithms, in terms of the number of nodes it expands. Determination of conditions under which MOA^* is optimal, or perhaps nondominated, if any such conditions exist, is an interesting research objective.

Another theoretical direction concerns the use of an imprecisely specified set of trade-off weights to direct the multiobjective search. This results in a parametric multiattribute formulation of MOA^* that should be much more tractable than the strictly multiobjective form described here. A further item of theoretical interest is the multiobjective generalization of the AO^* algorithm, the counterpart of A^* for use in searching AND/OR graphs.

The other, and perhaps more critical, basic direction for future research concerns the practical uses of multiobjective search. The preliminary work done so far indicates that these search routines will be very computationally intense. Issues of computational tractability and efficiency will certainly become critical for any practical applications of multiobjective search. Actual heuristics available for real applications will probably be hard to develop and admissibility may often be difficult to guarantee. Further work on searching with inadmissible heuristics may therefore be of practical significance. When combined with the computational advantages of the parametric multiattribute formulation of MOA^*, the incorporation of such inadmissible heuristics may make the algorithm tractable enough to be truly useful in practical multiobjective problem solving applications.

ACKNOWLEDGMENTS. This research has benefited greatly from numerous helpful comments that were provided by Douglas J. White, Dana Richards, Edward A. Sykes, and two anonymous reviewers.

REFERENCES

1. AHO, A. V., HOPCROFT, J. E., AND ULLMAN, J. D. *Data Structures and Algorithms*. Addison-Wesley, Reading, Mass., 1983.
2. BEHZAD, M., CHARTRAND, G., AND LESNIAK-FOSTER, L. *Graphs and Digraphs*. Prindle, Weber, and Schmidt, Boston, Mass., 1979.
3. BELLMAN, R. E. *Dynamic Programming*. Princeton Univ. Press, Princeton, N.J., 1957.
4. BOGETOFT, P. General communication schemes for multiobjective decision making. *Europ. J. Op. Res. 26* (1986), 108–122.
5. BROCKHOFF, K. Experimental test of MCDM algorithms in a modular approach. *Europ. J. Op. Res. 22* (1985), 159–166.
6. CANGPU, W. Multi-criteria dynamic programming. *Sci. Sinica 23* (July 1980), 814–822.
7. CARRAWAY, R. L., MORIN, T. L., AND MOSKOWITZ, H. Generalized dynamic programming for multicriteria optimization. *Europ. J. Op. Res. 44* (Jan. 1990), 95–104.
8. DONALD, B. R. A search algorithm for motion planning with six degrees of freedom. *Artif. Int. 31* (1987), 295–353.
9. HARARY, F. *Graph Theory*. Addison-Wesley, Reading, Mass., 1969.
10. 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* (1968), 100–107.
11. HART, P. E., NILSSON, N. J. AND RAPHAEL, B. "Correction to 'A formal basis for the heuristic determination of minimum cost paths.'" *SIGART Newsletter 37* (1972), 28–29.
12. HENIG, M. I. The principle of optimality in dynamic programming with returns in partially ordered sets. *Math. Op. Res. 10* (Aug. 1985), 462–470.
13. KEENEY, R. L. AND RAIFFA, H. *Decisions with multiple objectives: Preferences and value tradeoffs*. New York, 1976.
14. KOK, M. The interface with decision makers and some experimental results in interactive multiple objective programming methods. *Europ. J. Op. Res. 26* (1986), 96–107.
15. KOKSALAN, M., KARWAN, M. H., AND ZIONTS, S. Approaches for discrete alternative multiple criteria problems for different types of criteria. Working Paper No. 572, School of Management, SUNY Buffalo, Buffalo, N.Y., June 1983.

16. KORHONEN, P J A hierarchical interactive method for ranking alternatives with multiple qualitative criteria *Europ. J. Op. Res. 24* (1986), 265–276.
17. KORHONEN, P. J., MOSKOWITZ, H., AND WALLENIUS, J. A progressive algorithm for modeling and solving multiple-criteria decision problems *Op. Res. 34* (Sep–Oct. 1986), 726–731.
18. KWA, J. B H. On the consistency assumption, monotone criterion and the monotone restriction *SIGART Newsletter 103* (Jan. 1988), 29–31.
19. LEVINE, P. AND POMEROL, J. PRIAM, and interactive program for choosing among multiple attribute alternatives *Europ. J. Op. Res. 25* (1986), 272–280.
20. LIU, C. L. *Introduction to Combinatorial Mathematics*. McGraw-Hill, New York, 1968.
21. MOND, B AND ROSINGER, E E. Interactive weight assessment in multiple attribute decision making. *Europ. J. Op. Res. 22* (1985) 19–25.
22. NILSSON, N. J. *Problem-Solving Methods in Artificial Intelligence*. McGraw-Hill, New York, 1971
23. NILSSON, N. J. *Principles of Artificial Intelligence*. Morgan-Kaufmann, San Mateo, Calif., 1980.
24. PEARL, J. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, Reading, Mass., 1984.
25. SAWARAGI, Y., NAKAYAMA, H AND TANINO, T. *Theory of Multiobjective Optimization*. Academic Press, Orlando, 1985
26. STEWART, B. S. *Heuristic Search with a General Order Relation*. Ph.D. Dissertation, Univ. Virginia, Charlottesville, Va. Aug 1988.
27 TARJAN, R. A. *Data Structures and Network Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, Pa., 1983.
28. TUCKER, A. *Applied Combinatorics*. 2nd ed. John Wiley & Sons, New York, 1984.
29. VANSNICK, J. On the problem of weights in multiple criteria decision making the noncompensatory approach. *Europ. J. Op. Res. 24* (1986), 288–294.
30 WHITE, D. J. *Optimality and Efficiency*. Wiley, New York, 1982.
31. WHITE, C. C III, SAGE, A. P, AND DOZONO, S. A model of multiattribute decisionmaking and trade-off weight determination under uncertainty *IEEE Trans. Syst. Man. and Cyber. SMC-14* (Mar./Apr 1984), 223–229
32. WHITE, C C. III, STEWART, B. S., AND CARRAWAY, R. L. Multiobjective, preference-based search in acyclic OR-graphs. *Europ. J. Op. Res.* (1991), accepted for publication.
33. ZELENY, M *Multiple Criteria Decision Making*. McGraw-Hill, New York, 1982.

RECEIVED OCTOBER 1987; REVISED MARCH AND MAY 1990; ACCEPTED JUNE 1990