



Discrete Optimization

Multicriteria heuristic search [☆]

L. Mandow ^{*}, J.L. Pérez de la Cruz

Departamento Lenguajes y Ciencias de la Computación, ETSI Informática, Universidad de Málaga, P.O. Box 4114, 29080 Málaga, Spain

Received 7 July 2000; accepted 6 July 2002

Abstract

This paper extends the multicriteria decision paradigm to the heuristic search domain in a systematic way. A useful formal definition of multicriteria heuristic graph search problems is provided. Then the fundamental issues in multicriteria search are described in detail and a general framework is developed. Several new procedures are presented and analyzed for the usual multicriteria decision rules: multiobjective, multiattribute, goal satisfaction, and lexicographic search.

© 2002 Elsevier Science B.V. All rights reserved.

Keywords: Multiple criteria analysis; Search theory; Heuristic search

1. Introduction

Best-first search algorithms are a class of optimization algorithms for graph search problems. The uninformed case of best-first search was introduced by Dijkstra (1959) for problems with scalar additive cost measures. The informed or heuristic version is the celebrated A^* algorithm, developed by Hart et al. (1968, 1972). An important analysis of scalar best-first search algorithms was carried out by Pearl (1984). This excellent work included the development of several generalized best-first procedures, including BF^* for problems with general heuristic evaluation func-

tions, and Z^* for the particular class of problems with recursive heuristic cost measures. The A^* algorithm is, in fact, a particular case of Z^* . Pearl's work also includes detailed formal and empirical analysis of behavior for A^* under different assumptions, as well as a discussion on how heuristic information can be obtained.

In recent decades, the classical decision-making paradigm based on scalar optimization has been replaced in many settings by the more general and powerful paradigm of multiple-criteria decision making (Yu, 1985; Chankong and Haimes, 1983). Some important decision rules have caught the attention of researchers in this area. These are the multiobjective, multiattribute and goal-based decision-making frameworks. The lexicographic multiobjective framework, although less frequently used, will also be considered in this paper.

Although several researchers have felt the need to develop multicriteria best-first search algorithms, no systematic studies on general multicriteria

[☆] This work is partially supported by the Spanish national plan for R&D and FEDER funds under project 1fd97-1922.

^{*} Corresponding author. Tel.: +34-5-213-3302; fax: +34-5-213-1397.

E-mail addresses: lawrence@lcc.uma.es (L. Mandow), perez@lcc.uma.es (J.L. Pérez de la Cruz).

search have been carried out yet. Some fundamentals of this task were established by Loui (1983). His work addressed the development of an uninformed multiattribute search algorithm. Multiobjective versions of A^* have been also developed for the restricted case of trees (Navinchandra, 1991), and for the general case of graphs (Stewart and White, 1991). Applications of these algorithms are described in (Navinchandra, 1991; Sykes and White, 1991). The interesting work on generalized dynamic programming (GDP) by Carraway et al. (1990) has thus far resulted only in the highly limited U^* multiattribute search algorithm (White et al., 1992). However, the ideas behind GDP are also potentially relevant for general multicriteria heuristic search (MHS).

This paper addresses the task of developing a general framework for MHS algorithms. The first step is the formalization of the class of MHS problems. Then, the fundamental components for a general MHS procedure are presented in detail. An important contribution in this sense is POP^* , a general multicriteria search procedure that can be suitably instantiated for several particular classes of search problems. Another important contribution of this paper is the development of $POP-Z^*$, the multicriteria counterpart of the scalar Z^* algorithm. Different algorithms are obtained from $POP-Z^*$ for the different classes of multicriteria decision rules (multiobjective, multiattribute, goal-based, and lexicographic multiobjective). An important result of this work is that the particular contributions of previous research efforts in this field can be easily identified as particular cases of the proposed procedures. Furthermore, new multicriteria counterparts of A^* have also been easily developed for the first time for goal-based and lexicographic multiobjective problems. These are described elsewhere with applications (Fernández et al., 1999; Mandow et al., 1998). The multiobjective counterpart of A^* resulting from the work described in this paper also represents an interesting improvement over the MOA^* algorithm described in (Stewart and White, 1991).

Although the new algorithms proposed in this paper are certainly applicable to many domains, the authors' motivation for this research is the need to introduce multicriteria decision aid tools in

the domain of design problem solving. A description of the opportunities offered by different classes of MHS algorithms in design is presented in (Mandow, 1999; Mandow and Pérez de la Cruz, 2000).

The paper is organized as follows. Section 2 defines MHS problems. Section 3 presents POP^* , a new procedure for general preference-based search. The workings of this procedure are illustrated with an example and some interesting properties are proved in Section 4. Then, reasonable conditions for pruning are analyzed, and a useful kind of heuristic evaluation functions is presented. A family of multicriteria search procedures is described, and formal properties for some of these are presented in Section 7. The results obtained for POP^* are generalized in Section 8, where the search for all solutions to a problem is considered. Section 9 is devoted to practical issues in the selection and use of the proposed search algorithms. Finally, Sections 10 and 11 describe how previous works relate to the results presented in this paper, and some conclusions and future works are outlined.

2. Multicriteria heuristic search problems

Many important decision problems can be adequately modeled using a graph or network representation. For example, the *state space* representation conceptualizes problems by means of a discrete space of states linked by transitions that can be easily assimilated to a graph. Typical examples cover a wide range of applications, from pattern recognition to path planning or resource allocation. The use of graph search methods is ubiquitous nowadays in the solution of these kinds of problems. Some of these methods simply search for a *feasible solution*, while others are able to rank feasible solutions according to some preference and provide an *optimal solution*. Search techniques usually involve a heavy computational burden. However, heuristic search strategies that use some kind of additional (heuristic) information can reduce these computational costs for many problem instances. This section provides a formal definition for graph search problems that use multicriteria

preferences assuming that some heuristic information may be available. The following sections will be devoted to introducing a class of graph search procedures suitable for solving these problems.

A MHS problem consists in finding a path in a graph that is optimal according to a given set of criteria. The definition of a MHS problem presented here is an elaboration of the usual formal definition of graph search problems. A graph search problem is defined by a tuple (G, s, Γ) . G is a locally finite graph (i.e., only a finite number of arcs emanates from each node). Let $\text{NODES}(G)$ be the set of nodes in G . Then, $s \in \text{NODES}(G)$ is a start node, and $\Gamma \subseteq \text{NODES}(G)$ is a set of goal or destination nodes. A path P in G is a sequence of nodes (n_1, n_2, \dots, n_k) such that for every pair of consecutive nodes n_i, n_{i+1} , there is an arc in G from n_i to n_{i+1} . A solution to a search problem (G, s, Γ) , or feasible solution, is any acyclic path in G from node s to any node $\gamma \in \Gamma$. The set of feasible solutions to a problem will be denoted $P\text{-SET}(s, \Gamma)$. A partial solution path is any acyclic path in G starting from node s .

Frequently, graph search problems are not explicitly provided as a tuple (G, s, Γ) . A successor function $\text{SCS} : \text{NODES}(G) \rightarrow 2^{\text{NODES}(G)}$, that returns the set of nodes reached by arcs emanating from node n , may be provided instead of G . A predicate function $\text{destination-node} : \text{NODES}(G) \rightarrow \{\text{TRUE}, \text{FALSE}\}$, that is true only for destination nodes, may be provided instead of Γ . However, the notation (G, s, Γ) will be used regardless of the explicit or implicit nature of the problem.

A more general class of problems considers preferences over the set of feasible solutions. Preferences are statements concerning pairs of solutions that establish whether these are equally good for the decision maker (DM), or one is better than the other. In multicriteria decision theory, preferences are frequently established in the following way:

- (a) Let X be the set of feasible solutions to a problem.
- (b) A set of real attribute functions over solutions is selected: $y_1(x), y_2(x), \dots, y_q(x)$, where $\forall i, y_i : X \rightarrow \mathbb{R}$. An attribute vector $\mathbf{y}(x) =$

$(y_1(x), y_2(x), \dots, y_q(x))$ can be calculated then for each alternative, $\mathbf{y} : X \rightarrow \mathbb{R}^q$.

- (c) A binary relation “is better than” (denoted by \prec) is established over attribute vectors, so that if $\mathbf{y}(x_1) \prec \mathbf{y}(x_2)$, then alternative x_1 is preferable to alternative x_2 for the DM.

The solutions to a problem with preferences, or minimal solutions, are all feasible solutions with minimal attribute vectors. Given a set of vectors, $Y \subseteq \mathbb{R}^q$, and a binary relation, \prec , the set of optimal or minimal vectors in Y , $\text{MINIMAL}(Y, \prec)$, is defined as,

$$\text{MINIMAL}(Y, \prec) = \{\mathbf{y} \in Y / \nexists \mathbf{y}' \in Y, \mathbf{y}' \prec \mathbf{y}\}$$

Different kinds of preference relations can be obtained from the relation, \prec :

- \succ : “Is worse than”, the antagonistic of “is better than”, $\mathbf{y}_1 \succ \mathbf{y}_2 \iff \mathbf{y}_2 \prec \mathbf{y}_1$.
- \sim : “Is as good as”, $\mathbf{y}_1 \sim \mathbf{y}_2$ iff none of the relations $\mathbf{y}_1 \prec \mathbf{y}_2$ or $\mathbf{y}_1 \succ \mathbf{y}_2$ holds.
- \succsim : “Is as good or better than”, $\mathbf{y}_1 \succsim \mathbf{y}_2 \iff \mathbf{y}_1 \prec \mathbf{y}_2 \vee \mathbf{y}_1 \sim \mathbf{y}_2$.
- \succcurlyeq : “Is as good or worse than”, $\mathbf{y}_1 \succcurlyeq \mathbf{y}_2 \iff \mathbf{y}_1 \succ \mathbf{y}_2 \vee \mathbf{y}_1 \sim \mathbf{y}_2$.

To be consistent with the notion of preference, both \prec and \succ need to be irreflexive and asymmetric, and \sim needs to be reflexive and symmetric. According to Yu (1985), neither of \prec or \succ need to be transitive to induce a preference. However, all usual multicriteria preferences do define at least a partial order relation, i.e., \prec and \succ are transitive.

The set of all preference statements defined by the previous relations over a set of alternatives is called a preference structure. The notation (\mathbf{f}, \prec) or (F, \prec) will be used to denote a preference structure induced by the relation \prec over the set of vectors calculated for each alternative using function \mathbf{f} or F .

A MHS problem is defined by a tuple $(G, s, \Gamma, \mathbf{g}, \prec_g, F, \prec_f, E, \prec_e)$, where (G, s, Γ) denotes a graph search problem, and,

- (\mathbf{g}, \prec_g) induces a preference structure over feasible solution paths, where \mathbf{g} is an attribute

function $\mathbf{g} : P\text{-SET}(s, \Gamma) \rightarrow \mathbb{R}^q$ that associates an attribute vector to each feasible solution, and \prec_g is a binary relation that establishes preferences over these vectors.

- (F, \prec_f) induces a heuristic preference structure over partial solution paths, where F is a heuristic evaluation function $F : P\text{-SET}(s, \text{NODES}(G)) \rightarrow 2^{\mathbb{R}^r}$ that associates a finite set of heuristic vectors to each partial solution path, and \prec_f is a binary relation that establishes preferences over these vectors. The relation \prec_f will be read “is heuristically preferable to”.
- (E, \prec_e) induces a pruning preference structure over partial solution paths, where E is a heuristic function $E : P\text{-SET}(s, \text{NODES}(G)) \rightarrow 2^{\mathbb{R}^s}$ that associates a finite set of pruning vectors to each partial solution path, and \prec_e is a binary relation that establishes preferences over these vectors. The relation \prec_e will be read “allows eliminating (discarding or pruning)”. Frequently, the heuristic function E provides only a single pruning vector for each path. In such cases, the preference structure will be denoted by (\mathbf{e}, \prec_e) , where $\mathbf{e} : P\text{-SET}(s, \text{NODES}(G)) \rightarrow \mathbb{R}^s$.

The set of solutions to a problem $(G, s, \Gamma, \mathbf{g}, \prec_g, F, \prec_f, E, \prec_e)$, or minimal solutions, are all feasible solution paths with minimal attribute vectors according to (\mathbf{g}, \prec_g) . This set will be denoted $P\text{-SET}^*(s, \Gamma, \mathbf{g}, \prec_g)$. The set of the associated minimal attribute vectors will be denoted $G\text{-SET}^*(s, \Gamma, \mathbf{g}, \prec_g)$.

2.1. Example

Let’s consider the following example: a DM would like to plan a hike to a camping site in a nature reserve. Let the graph in Fig. 1 denote the forest trails in the reserve. The trip starts from node n_1 and finishes at the camping site located at node n_5 . The DM wishes to minimize the walking distance and the chances of coming upon a hungry bear on his way. Specifically, the DM would be willing to walk further to avoid entering zones populated with hungry bears according to some subjective multiattribute preference. The DM has a map of the reserve that indicates walking distances and the rate at which hikers have recently

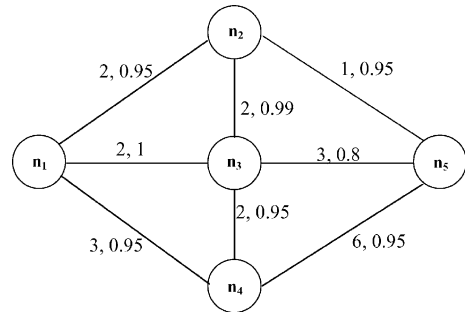


Fig. 1. A sample graph. Arcs are labeled with two values: (1) length (in kilometers); (2) probability that hungry bears will not be found.

met bears for each trail. However, this map provides the DM with some additional useful information: the straight-line distance from any point n to the destination n_5 can be easily estimated. This problem can be formally defined as follows:

- G is the graph in Fig. 1, all arcs are bi-directional, $s = n_1$ and $\Gamma = \{n_5\}$.
- *Preference:* Two attributes will be considered: $g_1(P) = \text{“length of path } P \text{ (in kilometers)”}$ and $g_2(P) = \text{“probability that hungry bears will not be met in path } P\text{”}$. Each arc in the graph is labeled with its length (in kilometers) and the associated probability. Attribute vectors for each path can be calculated as follows:

$$\mathbf{g}(P) = \mathbf{g}(s, n_1) \circ \mathbf{g}(n_1, n_2) \circ \dots \circ \mathbf{g}(n_{k-1}, n_k)$$

where $\mathbf{g}(n_i, n_j)$ is the attribute vector with the values of g_1 and g_2 associated with arc (n_i, n_j) , and $\circ = (+, \times)$ is the accumulative vector operator. It is assumed here that the probabilities associated with avoiding bears in each arc are independent, and therefore can be calculated multiplicatively for each path. A multiattribute preference over attribute vectors will be defined as follows:

$$\text{minimize } k(\mathbf{g}) = g_1 \times 10^{-g_2}$$

$$\text{More formally, } \forall \mathbf{g}, \mathbf{g}' \in \mathbb{R}^2, \mathbf{g} \prec_g \mathbf{g}' \iff k(\mathbf{g}) < k(\mathbf{g}')$$

The function $k(\mathbf{g})$ is intended to be some kind of overall difficulty measure for paths. Note that the first attribute is of the kind “the less the better”,

Table 1
Heuristic estimates to node n_5 (see graph in Fig. 1)

Node	$h(n)$ (kilometers, probability of no bears)
n_1	(2, 1)
n_2	(1, 1)
n_3	(1, 1)
n_4	(3, 1)
n_5	(0, 1)

while the second one is of the kind “the more the better”. For the sake of simplicity all of the discussion in this paper is centered around “the less the better” attributes. However, all the results can be easily extended to cover both kinds of attributes.

- *Heuristic preference:* A single heuristic evaluation vector will be calculated for each path:

$$\mathbf{f}(P) = \mathbf{g}(P) \circ \mathbf{h}(\text{last-node}(P))$$

where $\mathbf{h} : \text{NODES}(G) \rightarrow \mathbb{R}^2$ is a heuristic function that returns for each node an optimistic estimate of the distance and a bear avoidance probability to the destination node (see Table 1). Notice that 1 is always an optimistic estimate for the probability of avoiding bears. The heuristic preference will be the minimization of $k(\mathbf{f})$.

- *Pruning:* The attribute vectors will also be used as pruning vectors, i.e., $\mathbf{e}(P) = \mathbf{g}(P)$. The pruning preference will be the Pareto order applied to paths reaching the same node, i.e., given two paths P_1, P_2 such that $\text{last-node}(P_1) = \text{last-node}(P_2)$,

$$\begin{aligned} \mathbf{g}(P_1) \prec_e \mathbf{g}(P_2) &\iff g_1(P_1) \leq g_1(P_2) \wedge g_2(P_1) \\ &\geq g_2(P_2) \wedge \mathbf{g}(P_1) \neq \mathbf{g}(P_2) \end{aligned}$$

Section 7.2.2 shows that this is a good pruning criterion for this problem, since the minima of $k(\mathbf{g})$ are among the Pareto optima of attribute vectors.

3. A general preference-based search procedure

This section introduces POP* (Primer Orden de Preferencia, Spanish translation for ‘First in Order of Preference’), a general procedure for the resolution of MHS problems. This procedure can be used as a general architecture in the

development of search algorithms for particular classes of MHS problems as described in the following sections.

POP* uses a memory structure to record interesting paths or solution alternatives as well as important information about them (certain labels and associated attribute, heuristic, and pruning vectors, as described below). Paths in this memory are labeled either ‘open’ or ‘closed’. Paths that have not been considered for extension yet are ‘open’, while those that already have been are ‘closed’. The memory is consulted and updated at each iteration until a solution path is selected. The pseudocode of POP* is shown in Table 2.

The function $\text{last-node}(P)$ receives a path and returns its last node. Whenever $\text{last-node}(P)$ returns a destination node, then P is a feasible solution.

POP* is essentially a general best-first search procedure. However, it is not an algorithm in a formal sense until the memory structure and its operations are properly described (see below).

POP* tries to find a minimal solution path starting from node s . To achieve this, it iteratively and incrementally generates a set of partial solution paths, hoping that one of them will eventually be extended to reach a destination node. Three important features of this procedure are:

- Paths are not selected for extension randomly or arbitrarily. Paths are selected at each iteration (step 2) on a best-first basis when they have a heuristic estimate vector that is minimal according to (F, \prec_f) .
- Not all generated paths are kept in memory. Those with all their pruning vectors dominated according to (E, \prec_e) are eliminated.

Table 2
Pseudocode of POP*

1. Initialize-memory(s)
2. If not(open-paths-in-memory()), then return(FAILURE)
3. $P \leftarrow$ minimal-open-path-in-memory()
4. If destination-node(last-node(P)), then return(P)
5. For each node $n \in \text{SCS}(\text{last-node}(P))$. Extend-paths-in-memory(last-node(P), n)
6. Close-associated-paths(last-node(P))
7. Return to step 2

- Each path may be selected for extension only once. Let n be the last node in the selected path. All possible extensions from n are considered, and all paths leading to node n are extended at the same iteration.

3.1. The memory and its operations

The memory used by POP* stores, in fact, a set of acyclic paths or partial solutions. However, since all such paths emanate from the same start node, the set can be implemented in the form of a tree or an acyclic graph to save space. A variety of factors may influence design decisions regarding the actual implementation. It makes a difference whether the graph being searched is a tree, an acyclic graph or a general graph. It is also important whether the solution to be returned is a minimal solution path, or just the particular destination node that was reached.

Regardless of the actual implementation, several operations need to be performed by POP* on its memory. These can be classified and described as follows:

- (a) Constructors (operations that change the contents of memory):
- Initialize-memory(s): Creates a new memory structure where there is only one alternative or path recorded, starting at node s . Associated vectors are calculated and the path is labeled ‘open’.
 - Extend-paths-in-memory(n', n): All open paths in memory ending at node n' are recovered. The paths resulting from their extension to reach node n are considered for inclusion in the memory. This operation involves several important considerations that may result in the addition and/or elimination of paths from the memory. Cyclic paths are never included. But most important, uninteresting alternatives are eliminated by virtue of the pruning preference structure: only paths with at least a minimal pruning vector (according to the pruning preference) are kept in memory. This may prevent new alternatives from entering the memory, and/

or discard old ones. All new alternatives added to memory are labeled ‘open’, and their associated vectors are calculated.

- Close-associated-paths(n): All paths ending at node n change their label to ‘closed’.
- (b) Selectors (operations that do not change the contents of memory):
- Open-paths-in-memory(): Returns FALSE if no alternative in memory is labeled ‘open’, and TRUE otherwise.
 - Minimal-open-path-in-memory(): Returns one open path or alternative from memory with a minimal heuristic vector. An error is signaled if none are available. If several minimal alternatives are available, a tie-breaking procedure is used to choose among them (e.g., random selection, or feasible solutions first).

Note that some of these functions need problem-dependent data to carry out their tasks. Particularly, minimal-open-path-in-memory() uses the heuristic preference relation \prec_f ; initialize-memory(n) uses the functions E and F ; and extend-paths-in-memory(n', n) the pruning preference structure and heuristic function F .

Suitable management of memory can have a tremendous impact on overall search efficiency (measured in time and space). The design and implementation of suitable data structures and associated operators is an important research problem in itself that deserves serious attention.

Many standard solutions or “patterns” exist in the literature for this problem in the domain of scalar search, depending on problem needs. For example, several heapsort algorithms have been developed to efficiently carry out the operations over a list of ‘open’ alternatives.

A specific data structure for multiobjective search problems is used in MOA* (Stewart and White, 1991), though its management is mixed with the best-first logic of the algorithm. An analogous structure is presented in (Fernández et al., 1999), which represents an improvement over the one used for MOA*. However, the main focus of this paper is on the logic of multicriteria search and sufficient conditions for its success, which are independent of particular data struc-

Table 3
Trace of POP*

Iterations	Path selected	Paths considered (● = closed)	$g(P)$	$f(P)$ for open paths	$k(f)$ for open paths	Pruned?	Minimal f open paths (x)
0	–	n_1	(0, 1.0000)	(2, 1)	0.2		x
1	n_1	n_1 ●	(0, 1.0000)	–	–		–
		n_1-n_2	(2, 0.9500)	(3, 0.9500)	0.3366		–
		n_1-n_3	(2, 1.0000)	(3, 1.0000)	0.3000		x
		n_1-n_4	(3, 0.9500)	(6, 0.9500)	0.6732		–
2	n_1-n_3	n_1 ●	(0, 1.0000)	–	–		–
		n_1-n_2	(2, 0.9500)	(3, 0.9500)	0.3366		x
		n_1-n_3 ●	(2, 1.0000)	–	–		–
		n_1-n_4	(3, 0.9500)	(6, 0.9500)	0.6732		–
		$n_1-n_3-n_2$	(4, 0.9900)	(5, 0.9900)	0.5116		–
		$n_1-n_3-n_4$	(4, 0.9500)	–	–		x (by n_1-n_4)
		$n_1-n_3-n_5$	(5, 0.8000)	(5, 0.8000)	0.7924		–
3	n_1-n_2	n_1 ●	(0, 1.0000)	–	–		–
		n_1-n_2 ●	(2, 0.9500)	–	–		–
		n_1-n_3 ●	(2, 1.0000)	–	–		–
		n_1-n_4	(3, 0.9500)	(6, 0.9500)	0.6732		–
		$n_1-n_3-n_2$ ●	(4, 0.9900)	–	–		–
		$n_1-n_3-n_5$	(5, 0.8000)	(5, 0.8000)	0.7924		–
		$n_1-n_2-n_3$	(4, 0.9405)	(5, 0.9405)	0.5734		–
		$n_1-n_2-n_5$	(3, 0.9025)	(3, 0.9025)	0.3755		x
4	$n_1-n_3-n_2-n_5$		(5, 0.9405)	(5, 0.9405)	0.5734		–

tures used to record alternatives (provided their operations behave correctly according to the previous definitions, of course). Once the needs of MHS are properly established, particular data structures can be designed and evaluated.

3.2. Example

Table 3 summarizes a trace of POP* applied to the problem presented in Section 2.1. Several interesting features of POP* are illustrated in this simple example. The set of all paths considered for inclusion in memory is shown for each iteration together with their attribute and heuristic estimate vectors. Note that cyclic paths are never considered. Initially only path (n_1) is in memory. At iteration 1 it is selected and three new paths are stored in memory. At iteration 2 the path (n_1, n_3) is selected for expansion. Two different paths to node n_2 are then considered, as well as two different paths to node n_4 . In the first case neither of them

can be pruned, but in the second one the path (n_1, n_3, n_4) can be discarded in favor of (n_1, n_4). At iteration 3 the path (n_1, n_2) is selected. Since there is another open path (n_1, n_3, n_2) leading to node n_2 , it is also closed and its extensions are also considered and included in memory. At iteration 3 a solution path (n_1, n_3, n_5) is selected as the “best” open alternative and the procedure terminates. Note that if the search had been conducted without heuristic information, i.e. $\forall n, h(n) = (0, 1)$, the path (n_1, n_4) would also have been selected.

4. Properties of POP*

There exist several interesting properties that are desirable for any search algorithm. First of all, it is nice if an algorithm terminates whenever a solution exists. Second, it is important to terminate with a feasible solution. Finally, it is interesting to terminate giving an optimal solution. This section

proves these properties for POP* making very general assumptions about heuristic and pruning preferences. More practical details about pruning and heuristic preferences are presented in Sections 5 and 6, respectively. First, several useful definitions are introduced.

Definition 1. Given a path $P = (n_0, n_1, n_2, \dots, n_q)$, the set of partial solution subpaths of P , denoted $\text{PSSP}(P)$, is the set of $q + 1$ subpaths of P starting from n_0 . In other words, it is the set formed by all paths P_{0i} , $i = 0, \dots, q$, such that $P_{00} = (n_0)$, $P_{01} = (n_0, n_1), \dots, P_{0q} = P$.

Definition 2. An algorithm is 1-admissible if it terminates with a minimal solution path whenever at least one solution exists.

Definition 3. An algorithm is N -admissible if it terminates with the set of all minimal solution paths, whenever this set is finite, and at least one solution exists; or if it does not terminate when there is an infinite number of minimal solutions.

Definition 4. Let us consider a MHS problem $(G, s, \Gamma, \mathbf{g}, \prec_g, F, \prec_f, E, \prec_e)$.

- The heuristic preference structure (F, \prec_f) is *optimistic* whenever for each minimal solution $P \in P\text{-SET}^*(s, \Gamma, \mathbf{g}, \prec_g)$ there is a heuristic estimate $\mathbf{f}^* \in F(P)$ such that,

$$\forall P_{0i} \in \text{PSSP}(P) \exists \mathbf{f}' \in F(P_{0i}) / \mathbf{f}' \prec_f \mathbf{f}^* \vee \mathbf{f}' = \mathbf{f}^*$$

- The heuristic preference structure (F, \prec_f) *agrees with* (\mathbf{g}, \prec_g) if all feasible solution paths have only a single heuristic estimate vector, and for any two feasible solutions, P', P'' , with estimates $F(P') = \{\mathbf{f}'\}$ and $F(P'') = \{\mathbf{f}''\}$, holds that,

$$\mathbf{f}' \prec_f \mathbf{f}'' \iff \mathbf{g}(P') \prec_g \mathbf{g}(P'')$$

- The pruning preference (E, \prec_e) is *1-cautious according to* (\mathbf{g}, \prec_g) whenever for each minimal attribute vector $\mathbf{g}^* \in G\text{-SET}^*(s, \Gamma, \mathbf{g}, \prec_g)$, there is at least one solution path P , such that $\mathbf{g}(P) = \mathbf{g}^*$, and all its partial solution subpaths have at least a pruning vector that is minimal according to (E, \prec_e) . The subset of minimal so-

lutions that makes this condition true will be called the set of reachable minimal solutions.

Lemma 1. *If (E, \prec_e) is 1-cautious according to (\mathbf{g}, \prec_g) and at least one solution exists, then for each reachable minimal solution P^* , there is always one open path in memory $P_{0i} \in \text{PSSP}(P^*)$ before termination.*

Proof. By induction on the number of iterations. The path $P_{00} = (s)$ is open in memory in the beginning. Let $P_{0i} = (s, \dots, n_i)$, $P_{0i} \in \text{PSSP}(P^*)$ be an open path in memory at iteration t_k . Since P_{0i} has a minimal pruning vector according to (E, \prec_e) it will never be eliminated from memory. Therefore, if at iteration t_{k+1} a path leading to a node $n \neq n_i$ is selected, P_{0i} will remain open. If a path leading to node n_i is selected, then all extensions of P_{0i} will be considered for inclusion in memory. One of them will be $P_{0i+1} = (s, \dots, n_i, n_{i+1})$, $P_{0i+1} \in \text{PSSP}(P^*)$. Since P_{0i+1} has also a minimal pruning vector according to (E, \prec_e) , it will be certainly included and opened. \square

Lemma 2. *If (E, \prec_e) is 1-cautious according to (\mathbf{g}, \prec_g) and at least one solution exists, then POP* cannot terminate returning FAILURE.*

Proof. Trivial from the proof of Lemma 1. \square

Proposition 1. *POP* always terminates on finite graphs.*

Proof. When G is finite, there is only a finite amount of partial solution paths. At each iteration POP* closes at least one path. Each path can be opened (i.e., added to memory) only once, so in the worst case all would be examined and closed in a finite number of iterations, and POP* would terminate returning FAILURE. \square

Proposition 2. *POP* is 1-admissible for finite graphs under the following (sufficient) conditions:*

1. *The heuristic preference structure (F, \prec_f) is optimistic.*
2. *The heuristic preference structure (F, \prec_f) agrees with (\mathbf{g}, \prec_g) .*

3. The heuristic preference \prec_f is transitive.
4. The pruning preference (E, \prec_e) is 1-cautious according to (\mathbf{g}, \prec_g) .

Proof. Let us assume at least one solution exists. Proposition 1 guarantees that POP* will terminate, and condition 4 guarantees that POP* will not terminate returning FAILURE (Lemma 2). It suffices to prove that POP* terminates with a minimal solution according to (\mathbf{g}, \prec_g) . However, by condition 2 this amounts to proving that POP* terminates with a minimal solution according to (F, \prec_f) .

Since (F, \prec_f) is optimistic, then for each reachable minimal solution path P^* holds that,

$$F(P^*) = \{\mathbf{f}^*\} \wedge \forall P_{0i} \in \text{PSSP}(P^*) \exists \mathbf{f}_{0i} \in F(P_{0i}) / \mathbf{f}_{0i} \preceq_f \mathbf{f}^*$$

Now, by virtue of conditions 2 and 4, for each dominated solution path P' , there exists at least one reachable minimal solution path P^* such that,

$$F(P^*) = \{\mathbf{f}^*\} \wedge F(P') = \{\mathbf{f}'\} \wedge \mathbf{f}^* \prec_f \mathbf{f}'$$

Since \prec_f satisfies the transitive property, then it will also hold that,

$$\forall P_{0i} \in \text{PSSP}(P^*) \exists \mathbf{f}_{0i} \in F(P_{0i}) / \mathbf{f}_{0i} \preceq_f \mathbf{f}^* \prec_f \mathbf{f}'$$

Since one subpath $P_{0i} \in \text{PSSP}(P^*)$ is always open before termination, then POP* can never select a dominated solution path in step 3. So the only remaining option is that POP* is 1-admissible. \square

Proposition 3. POP* is 1-admissible for infinite graphs when, in addition to conditions 1, 2, 3, and 4, the following (sufficient) condition holds,

5. For each minimal solution path P^* with $F(P^*) = \{\mathbf{f}^*\}$, there exists only a finite number of partial solution paths P' such that,

$$\exists \mathbf{f}' \in F(P') / (\mathbf{f}' \preceq_f \mathbf{f}^* \vee \mathbf{f}' = \mathbf{f}^*)$$

Proof. Note that condition 5 implies, by definition, that $P\text{-SET}^*(s, \Gamma, \mathbf{g}, \prec_g)$ is finite. The proof is quite trivial. Assume at least one solution exists. Then condition 5 guarantees that the number of partial solution paths with estimates preferable to

any minimal solution will also be finite. Therefore, in the worst case, POP* will terminate selecting a minimal solution after selecting in a finite number of steps all those partial solutions. \square

5. Conditions for cautious pruning

Only very general notions have been provided so far on the relationship between the pruning preference (E, \prec_e) and the problem's preference (\mathbf{g}, \prec_g) . However, the role of pruning in search should not be overlooked. Good pruning preferences constrain the search and hence result in more efficient procedures. This section reviews the 'order-preserving' property used in many scalar optimization algorithms, and provides operational conditions that guarantee cautious pruning in multicriteria problems.

5.1. Order-preserving property

The order-preserving property (Pearl, 1984, p. 102) can be defined as follows.

Definition 5. Let $P_i P_j$ denote the concatenation of two paths. A real function $e(P)$ is order-preserving whenever for any set of paths $P_i, i = 1, 2, 3$ in a graph G such that $P_1 P_3$ and $P_2 P_3$ are also paths in G , the following holds,

$$e(P_1) \leq e(P_2) \Rightarrow e(P_1 P_3) \leq e(P_2 P_3)$$

Some simple scalar functions, like the additive cost measure used with A^* , straightforwardly satisfy this condition, and therefore can be used also as pruning criteria. Some scalar best-first algorithms like A^* or Dijkstra's algorithm use the problem's preference $(g, <)$ also as pruning preference. However, Loui (1983) showed that the analogous solution is not always possible in multicriteria search.

The following simpler definitions will be useful in the next sections.

Definition 6. Let $P_i, i = 1, 2, 3$ be any set of paths in a graph G such that $P_1 P_3$ and $P_2 P_3$ are also paths in G .

- A real function $e(P)$ preserves the order $<$ if,
 $e(P_1) < e(P_2) \Rightarrow e(P_1P_3) < e(P_2P_3)$
- A real function $e(P)$ preserves the order \leq if,
 $e(P_1) < e(P_2) \Rightarrow e(P_1P_3) \leq e(P_2P_3)$
- A real function $e(P)$ preserves equality if,
 $e(P_1) = e(P_2) \Rightarrow e(P_1P_3) = e(P_2P_3)$

5.2. Multicriteria order preservation

Multicriteria order preservation is a multicriteria generalization of the order-preserving property used in scalar algorithms like BF^* , Z^* or A^* , and is another contribution of this paper.

Definition 7. Let $(G, s, \Gamma, \mathbf{g}, \prec_g, F, \prec_f, \mathbf{e}, \prec_e)$ be a MHS problem (note that a vector function $\mathbf{e}(P)$ is used for pruning). Let P_iP_j denote the concatenation of two paths, and $P_i, i = 1, 2, 3$ any set of paths in G such that P_1P_3 and P_2P_3 are also paths in G .

- (a) The pruning preference (\mathbf{e}, \prec_e) preserves the preference (\mathbf{g}, \prec_g) iff,
 $\mathbf{e}(P_1) \prec_e \mathbf{e}(P_2) \Rightarrow \mathbf{g}(P_1P_3) \prec_g \mathbf{g}(P_2P_3)$
- (b) The pruning preference (\mathbf{e}, \prec_e) preserves or is richer than the preference (\mathbf{g}, \prec_g) iff,
 $\mathbf{e}(P_1) \prec_e \mathbf{e}(P_2) \Rightarrow \mathbf{g}(P_1P_3) \preceq_g \mathbf{g}(P_2P_3)$
- (c) The pruning preference (\mathbf{e}, \prec_e) preserves equality with (\mathbf{g}, \prec_g) iff,
 $\mathbf{e}(P_1) = \mathbf{e}(P_2) \Rightarrow \mathbf{g}(P_1P_3) = \mathbf{g}(P_2P_3)$

Note that preference statements in the previous definition apply only to pairs of paths leading to the same node. The pruning preference between paths leading to different nodes is undefined.

Proposition 4. Given a problem $(G, s, \Gamma, \mathbf{g}, \prec_g, F, \prec_f, \mathbf{e}, \prec_e)$, if (\mathbf{e}, \prec_e) preserves or is richer than (\mathbf{g}, \prec_g) , then it is also 1-cautious.

Proof. Let P^* be any minimal solution path, and $P_{0i} = (s, \dots, n_i) \in \text{PSSP}(P^*)$. Let us assume P_{0i} is pruned. Then, there must be some partial solution

path P' leading to n_i , such that, $\mathbf{e}(P') \prec_e \mathbf{e}(P_{0i})$. Now, let P'' be an extension of P_{0i} such that $P^* = P_{0i}P''$. Since (\mathbf{e}, \prec_e) preserves or is richer than (\mathbf{g}, \prec_g)

$$\mathbf{e}(P'P'') \prec_e \mathbf{e}(P^*) \Rightarrow \mathbf{g}(P'P'') \preceq_g \mathbf{g}(P^*)$$

However, the relation $\mathbf{g}(P'P'') \prec_g \mathbf{g}(P^*)$ contradicts the assumption that P^* is a minimal solution. Therefore, a minimal solution path P^* can only be discarded by another minimal solution path P^{**} such that $\mathbf{g}(P^*) = \mathbf{g}(P^{**})$. Consequently, the pruning is 1-cautious. \square

Proposition 5. Let $(G, s, \Gamma, \mathbf{g}, \prec_g, F, \prec_f, \mathbf{e}, \prec_e)$ be a problem where (\mathbf{e}, \prec_e) preserves equality with (\mathbf{g}, \prec_g) . Let P, P' be two partial solution paths leading to the same node with $\mathbf{e}(P) = \mathbf{e}(P')$. If any of P, P' is discarded, then the resulting pruning is 1-cautious.

Proof. Trivial from the proof of the previous proposition. \square

In the light of the previous result, the following definition formalizes a pruning preference frequently used in 1-admissible graph search algorithms.

Definition 8. Assume that two different paths P_1, P_2 in a graph reach the same node, and $e(P_1) = e(P_2)$. If (\mathbf{e}, \prec_e) preserves equality with (\mathbf{g}, \prec_g) , then pruning one of both paths still ensures 1-cautiousness (see Proposition 5). Any arbitrary pruning criterion used in this cases will be called equality tie-breaking.

The usual equality tie-breaking procedure is to keep the path that was generated first during the search and prune the newly generated one. However, any other arbitrary tie-breaking procedure will also work.

6. Cumulative attributes and heuristic evaluation functions

This section focuses attention on a particular class of search problems; those with cumulative

attribute functions. These problems allow efficient computation of attribute vectors as well as a simple and effective kind of heuristic estimates.

6.1. Cumulative attributes

Cumulative attribute functions can be easily calculated in best-first search algorithms.

Definition 9. A real attribute function $g_i(P)$ is cumulative if it can be calculated for any path $P = (n_1, n_2, \dots, n_k)$ using an associative combining operator \circ_i in the following way,

$$g_i(P) = E(n_1) \circ_i E(n_2) \circ_i \dots \circ_i E(n_k)$$

where $E(n)$ is some local property of node n , i.e., the cost associated to the arc arriving to (or emanating from) node n .

A multicriteria search problem is cumulative if all its attributes are cumulative.

The importance of these problems stems from the fact that many usual combining functions are cumulative, e.g., addition (+), product (\times), maximum (max), or minimum (min).

6.2. Evaluation functions for cumulative attributes

Cumulative attributes offer a simple and natural way to combine with a usual kind of heuristic information.

Let $h_i(n)$ be a heuristic function that returns an estimate of the value of attribute $g_i(P)$ for a path in $P\text{-SET}(n, \Gamma)$. Then an estimate of attribute $g_i(P)$ for an extension of a partial solution path $P' = (n_1, n_2, \dots, n_k)$ to a goal node can be easily calculated as follows:

$$\begin{aligned} E(n_1) \circ_i E(n_2) \circ_i \dots \circ_i E(n_k) \circ_i h_i(n_k) \\ = g_i(P) \circ_i h_i(n_k) \end{aligned}$$

The following definitions extend these concepts to multicriteria problems with cumulative attribute vectors.

Definition 10. Let $(G, s, \Gamma, \mathbf{g}, \prec_g, F, \prec_f, E, \prec_e)$ be a MHS problem with cumulative attributes. A multicriteria heuristic function $H: \text{NODES}(G) \rightarrow 2^{\mathbb{R}^q}$

is a function that for each node in G returns a finite set of vectors, that are attribute estimates of the paths in $P\text{-SET}(n, \Gamma)$.

Definition 11. Let $(G, s, \Gamma, \mathbf{g}, \prec_g, F, \prec_f, E, \prec_e)$ be a MHS problem with cumulative attributes. A heuristic evaluation function $F(P)$ is *direct* if it is calculated using a multicriteria heuristic function $H(n)$ in the following way:

$$F(P) = \{\mathbf{f}/\mathbf{f} = \mathbf{g}(P) \circ \mathbf{h} \wedge \mathbf{h} \in H(\text{last-node}(P))\}$$

where $\circ = (\circ_1, \circ_2, \dots, \circ_q)$ is the vector combining function, i.e., each \circ_i is the combining operator of the i th attribute in $\mathbf{g}(P)$.

In short, the function $F(P)$ may provide one or more attribute estimates for one or more extensions of path P to destination nodes.

6.3. Conditions for direct evaluation functions

Definition 12. Let $(G, s, \Gamma, \mathbf{g}, \prec_g, F, \prec_f, E, \prec_e)$ be a problem where \mathbf{g} is cumulative with operator \circ , and F is direct. A multicriteria heuristic function $H(n)$ is *admissible* for the problem when the following conditions hold:

- $\forall \gamma \in \Gamma, H(\gamma) = \{\mathbf{n}\}$, where \mathbf{n} is the neutral element of operator \circ , i.e.,

$$\forall \mathbf{v} \in \mathbb{R}^q, \quad \mathbf{v} \circ \mathbf{n} = \mathbf{v}$$
- $\forall P^* \in P\text{-SET}^*(s, \Gamma, \mathbf{g}, \prec_g)$ with $F(P) = \{\mathbf{f}^P\} \wedge \forall P_{0i} \in \text{PSSP}(P^*)$,

$$\exists \mathbf{h} \in H(\text{last-node}(P_{0i}))/\mathbf{g}(P_{0i}) \circ \mathbf{h} \preceq_f \mathbf{f}^P$$

Proposition 6. Let $(G, s, \Gamma, \mathbf{g}, \prec_g, F, \prec_f, E, \prec_e)$ be a problem where \mathbf{g} is cumulative with operator \circ , and F is direct. If an admissible heuristic function is used, then,

- (F, \prec_f) agrees with (g, \prec_g) .
- (F, \prec_f) is optimistic.

Proof. In fact, the definition of an admissible heuristic function (see Definition 12) is a more operational reformulation of conditions (a) and (b) for these particular kinds of problems,

- (a) It is rather simple to show that (F, \prec_f) agrees with (g, \prec_g) ,

$$\forall \gamma \in \Gamma, \quad H(\gamma) = \{\mathbf{n}\} \Rightarrow \forall P \in P\text{-SET}(s, \Gamma), \\ F(P) = \{\mathbf{g}(P) \circ \mathbf{n} = \mathbf{g}(P)\}$$

- (b) It is easy to prove that (F, \prec_f) is optimistic. From the previous result,

$$\forall P^* \in P\text{-SET}^*(s, \Gamma, \mathbf{g}, \prec_g), \\ F(P) = \{\mathbf{f}^P = \mathbf{g}(P)\}$$

Now, by definition of an admissible multicriteria heuristic function,

$$\forall P^* \in P\text{-SET}^*(s, \Gamma, \mathbf{g}, \prec_g), \quad \forall P_{0i} \in \text{PSSP}(P^*) \\ \exists \mathbf{h} \in H(\text{last-node}(P_{0i})/\mathbf{g}(P_{0i})) \circ \mathbf{h} \preceq_f \mathbf{f}^P$$

And, also by definition $\mathbf{g}(P_{0i}) \circ \mathbf{h} \in F(P_{0i})$. Therefore, (F, \prec_f) is optimistic. \square

7. Formal properties for some new search procedures

This section combines the results presented in previous sections to obtain a set of useful MHS procedures. All the procedures discussed in this section are obtained as instances of a procedure called POP-Z*, that results from a triple instantiation of POP*:

- (a) First, (\mathbf{g}, \prec_g) is limited to cumulative attribute functions.
- (b) Then, (F, \prec_f) is limited to direct evaluation functions, and the preference relation $\prec_f \equiv \prec_g$ (see Section 6).
- (c) Finally, the pruning preference (\mathbf{e}, \prec_e) preserves or is richer than (\mathbf{g}, \prec_g) and preserves equality with (\mathbf{g}, \prec_g) (see Section 5). Normally, \mathbf{e} is the function \mathbf{g} , i.e., no heuristic information is used for pruning.

The result is a simple and powerful procedure that can be further instantiated for the usual multicriteria decision rules: multiobjective, lexicographic, multiattribute, and goal satisfaction. It is not difficult to obtain sufficient conditions for these decision rules that imply the assumptions of POP-Z* and are therefore 1-admissible.

For each preference rule, two different local pruning preferences have been tested,

- (a) The preference relation \prec_g with equality tie-breaking. This is usual in scalar best-first algorithms with additive attributes, but unfortunately it is not suitable for many multicriteria problems.
- (b) The Pareto order with equality tie-breaking. This was proposed by Loui (1983) for multiattribute blind search, but turns out to be useful for other multicriteria decision rules as well.

Note that pruning preferences will be established only between paths leading to the same node.

The result are six different procedures (summarized in Table 4) that are 1-admissible under different conditions. Four of them can be further instantiated for the case of additive attributes, and result in different multicriteria counterparts of A*.

All these procedures are shown in Fig. 2. Procedures using Pareto order as pruning preference are labeled with subscript L (e.g., METAL-Z_L*, Z_L*). Those using \prec_g as pruning preference do not carry subscript (e.g., METAL-Z*, Z*).

7.1. Multiobjective search

The preference relation in multiobjective problems usually involves the minimization of multiple attributes. This is the so-called Pareto order. Given a set of alternatives X , and a set of attribute functions $g_i(x)$, it is usually stated as,

$$\forall i \text{ minimize } g_i(x)$$

More formally, this preference relation can be defined as follows:

$$\forall \mathbf{g}, \mathbf{g}' \in \mathbb{R}^q, \quad \mathbf{g} \prec_g \mathbf{g}' \iff \forall i \ g_i \leq g'_i \wedge \mathbf{g} \neq \mathbf{g}'$$

where g_i denotes the i th component of vector \mathbf{g} .

7.1.1. Procedure MO-Z*

Since the problems' preference is also the Pareto order, only one pruning preference has been considered for multiobjective problems: Pareto order with equality tie-breaking. The resulting procedure is called MO-Z* (Multi-Objective Z*).

Table 4
Some instances of POP-Z* according to different problem and pruning preferences

Pruning preferences (paths leading to the same node)	Problem preferences			
	Multiobjective	Multiattribute	Lexicographic	Goal satisfaction
\prec_g		Z^*	PRIMO-Z*	–
Pareto order	MO-Z*	Z_L^*	PRIMO- Z_L^*	METAL- Z_L^*

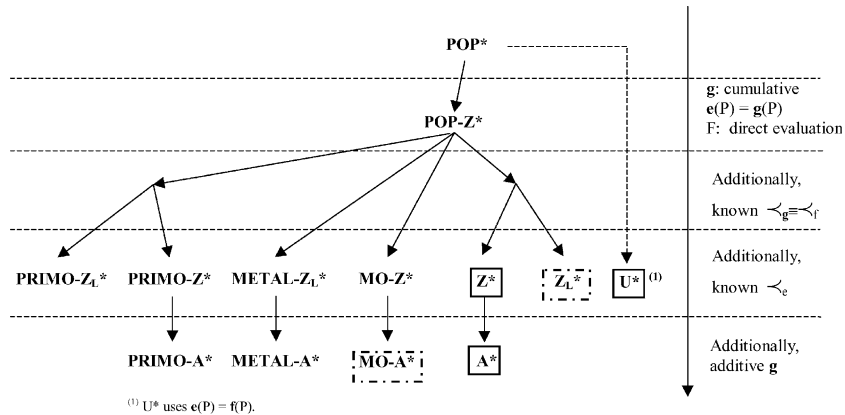


Fig. 2. A family of multicriteria search procedures. Algorithms previously described in the literature are shown inside squares. MO-A* and Z_L^* extend or improve previous algorithms.

Proposition 7. MO-Z* is 1-admissible on finite graphs if,

- (a) $H(n)$ is admissible.
- (b) Each attribute $g_i(P)$ preserves equality and one of $<$ or \leq .

It is also 1-admissible for infinite graphs if additionally,

- (c) Estimate vectors in $F(P)$ are not bounded in any of their components for infinite paths.

Proof. Let us check whether the admissibility conditions of POP* are satisfied:

- Admissibility of $H(n)$ implies conditions 1 and 2 (see Proposition 6).
- The Pareto order is trivially transitive (condition 3).
- Let us examine condition 4. It suffices to prove that condition (b) implies that (\mathbf{g}, \prec_g) preserves or is richer than itself, which by proposition 4 implies it is 1-cautious. Let G be a graph, and

$P_i, i = 1, 2, 3$ any set of paths in G such that P_1P_3 and P_2P_3 are also paths in G . Let us assume $\mathbf{g}(P_1) \prec_g \mathbf{g}(P_2)$, that is,

$$\forall i \ g_i(P_1) \leq g_i(P_2) \wedge \exists j \ g_j(P_1) < g_j(P_2)$$

When each $g_i(P)$ preserves equality and one of \leq or $<$, then

$$\forall k \ g_k(P_1) = g_k(P_2) \Rightarrow g_k(P_1P_3) = g_k(P_2P_3)$$

$$\forall m \ g_m(P_1) < g_m(P_2) \Rightarrow g_m(P_1P_3) \leq g_m(P_2P_3)$$

In consequence,

$$\forall i \ g_i(P_1P_3) \leq g_i(P_2P_3)$$

and (\mathbf{g}, \prec_g) preserves or is richer than itself. Note that this proof is independent of the cumulative nature of the attributes.

- Finally, let us examine condition 5, regarding infinite graphs. Let $\mathbf{f}^* = (f_1^*, f_2^*, \dots, f_q^*)$ be the heuristic estimate of any reachable minimal solution path P^* . Condition (c) implies that there can only be a finite number of partial solution paths with estimates $\mathbf{f}' = (f'_1, f'_2, \dots, f'_q)$ preferable equal or indifferent to \mathbf{f}^* , i.e., such that,

$$\exists i f'_i \leq f_i^*$$

Therefore, P^* will be selected after all those paths in a finite number of steps. \square

It can be trivially shown that functions like addition, multiplication (constrained to non-negative attributes), calculating the maximum of a set, and calculating the minimum of a set, preserve $<$ or \leq , and equality, and therefore can be used safely in multicriteria problems solved with MO- Z^* .

7.2. Multiattribute search

The preference relation in multiattribute problems usually involves the minimization of a scalar cost function $c(P)$ (or, more precisely, $c(\mathbf{g}(P))$), that combines several attributes from each alternative P . This preference can be formally stated as follows:

$$\forall \mathbf{g}, \mathbf{g}' \in \mathbb{R}^q, \quad \mathbf{g} \prec_g \mathbf{g}' \iff c(\mathbf{g}) < c(\mathbf{g}')$$

This kind of problem has important similarities with scalar optimization problems. Minimal solutions are the optima of $c(\mathbf{g})$. This function induces a weak order relation between attribute vectors. Therefore, it is possible to think of each path as having a single scalar cost estimate,

$$k(P) = \min_{\mathbf{f} \in F(P)} \{c(\mathbf{f})\}$$

Two different procedures, Z^* and Z_L^* , are analyzed here for this kind of problem.

7.2.1. Procedure Z^*

The use of $\prec_e \equiv \prec_g$ and equality tie-breaking yields a procedure that is essentially similar to the scalar optimization algorithm Z^* (Pearl, 1984). Therefore the name has been preserved.

Proposition 8. Z^* is 1-admissible for finite graphs if,

- (a) $H(n)$ is admissible.
- (b) The cost function $c(P)$ is order-preserving (i.e., preserves \leq and equality).

It is also 1-admissible for infinite graphs if additionally,

- (c) The function $k(P)$ is not bounded for infinite paths.

Proof. Let us check that the admissibility conditions of POP* are satisfied:

- Admissibility of $H(n)$ implies conditions 1 and 2 (see Proposition 6).
- The multiattribute preference is a weak order and, in consequence, transitive (condition 3).
- Let us consider condition 4. Since \prec_g is order-preserving, it is also trivially 1-cautious.
- Finally, let us consider condition 5. Let k^* be the optimum value of $k(P)$ reached by all minimal solutions. Condition (c) implies that there can only be a finite number of partial solution paths P' such that $k(P') < k^*$. Therefore, a minimal solution will be selected in the worst case after all these paths in a finite number of steps. \square

Apparently, there are only two multiattribute cost functions that are order-preserving, and both are cumulative (Carraway et al., 1990):

- (a) All attributes $g_i(P)$ additive, and $c(\mathbf{g}) = \kappa_1 g_1 + \kappa_2 g_2 + \dots + \kappa_q g_q, \forall \kappa_i \in \mathbb{R}$.
- (b) All attributes $g_i(P)$ multiplicative and non-negative, and,

$$c(\mathbf{g}) = \kappa g_1^{\kappa_1} g_2^{\kappa_2} \dots g_q^{\kappa_q}, \quad \text{where } \kappa \in \mathbb{R}.$$

Therefore it does not make sense to seek a multicriteria generalization of BF* (Pearl, 1984) where the cumulative requirement for cost functions is relaxed, but order-preservation is still required. In fact, POP* is a multicriteria generalization of BF* where the pruning preference was kept intentionally generic. Of course, the scalar BF* is also a particular case of POP*.

7.2.2. Procedure Z_L^*

In Z_L^* the pruning preference is the Pareto order with equality tie-breaking.

Definition 13. A cost function $c(\mathbf{g})$ is monotonous with attribute vectors if,

- (a) All attributes g_i are either of the kind “the more the better” or “the less the better”.
- (b) If an attribute g_i is of the kind “the more the better”, then, for any two vectors $\mathbf{g}, \mathbf{g}' \in \mathbb{R}^q$,

$$\exists i \ g_i > g'_i \wedge \forall j \neq i \ g_j = g'_j \Rightarrow c(\mathbf{g}) < c(\mathbf{g}')$$
- (c) If an attribute g_i is of the kind “the less the better”, then, for any two vectors $\mathbf{g}, \mathbf{g}' \in \mathbb{R}^q$,

$$\exists i \ g_i < g'_i \wedge \forall j \neq i \ g_j = g'_j \Rightarrow c(\mathbf{g}) < c(\mathbf{g}')$$

Proposition 9. Z_L^* is 1-admissible for finite graphs if,

- (a) $H(n)$ is admissible.
- (b) The cost function $c(\mathbf{g})$ is monotonous with attribute vectors.
- (c) All attribute functions are order-preserving (i.e., preserve \leq and equality).

It is also 1-admissible for infinite graphs if additionally,

- (d) The heuristic function $k(P)$ is not bounded for infinite paths.

Proof. Let us check that the admissibility conditions of POP* are satisfied:

- The proofs for conditions 1, 2, 3 and 5 are identical to those presented for the Z^* algorithm (see Section 7.2.1).
- Let us examine condition 4. Since $c(\mathbf{g})$ is monotonous, then the optima of $c(\mathbf{g})$ are among the Pareto optima of $\{\mathbf{g}(P)/P \in P\text{-SET}(s, \Gamma)\}$. Since all $g_i(P)$ are order-preserving, then the pruning of Pareto optima will be 1-cautious, and consequently, the pruning of $c(\mathbf{g})$ optima will be 1-cautious too. \square

Procedure Z_L^* is, in fact, a heuristic version of the procedure described by Loui (1983).

7.3. Lexicographic search

Lexicographic preference (see for example Chankong and Haimes, 1983, pp. 200–205) is based on the idea of sequential elimination of al-

ternatives. It can be easily stated as: minimize $g_1(x)$; then minimize $g_2(x)$ among the minima obtained in the previous step; then minimize $g_3(x)$ among the minima obtained in the previous step; and so on. More formally, this preference can be stated as,

$$\forall \mathbf{g}, \mathbf{g}' \in \mathbb{R}^q, \quad \mathbf{g} \prec_g \mathbf{g}' \iff \exists j \ g_j < g'_j \wedge \forall i < j \ g_i = g'_i$$

Two new procedures are analyzed for these kinds of problems: PRIMO- Z^* and PRIMO- Z_L^* (PRIoritized Multiple-Objective Z^*).

7.3.1. Procedure PRIMO- Z^*

PRIMO- Z^* uses the lexicographic preference relation with equality tie-breaking as the pruning preference.

Proposition 10. PRIMO- Z^* is 1-admissible for finite graphs if,

- (a) $H(n)$ is admissible.
- (b) All attribute functions preserve $<$ and equality.

It is also 1-admissible for infinite graphs if additionally,

- (c) Estimate vectors in $F(P)$ are not bounded in any of their components for infinite paths.

Proof. Let us check that the admissibility conditions of POP* are satisfied:

- Admissibility of $H(n)$ implies both conditions 1 and 2 (see Proposition 6).
- The lexicographic preference is a strict order and, in consequence, transitive (condition 3).
- Condition 4 is proved as follows. Let us assume $\mathbf{g} \prec_g \mathbf{g}'$, that is, $\exists j \ g_j < g'_j \wedge \forall i < j \ g_i = g'_i$. Now, since $g_j(P)$ preserves $<$, and all the remaining attributes preserve equality,

$$\forall \mathbf{v} \in \mathbb{R}^q, \quad \mathbf{g} \circ \mathbf{v} \prec_g \mathbf{g}' \circ \mathbf{v} \Rightarrow \exists j \ g_j \circ_j v_j < g'_j \circ_j v_j \wedge \forall i < j \ g_i \circ_i v_i = g'_i \circ_i v_i$$

Therefore pruning will be 1-cautious.

- The proof of condition 5, regarding infinite graphs, is also very simple. Let $\mathbf{f}^* =$

$(f_1^*, f_2^*, \dots, f_q^*)$ be the heuristic estimate of any reachable minimal solution path P^* . Condition (c) implies that there can only be a finite number of partial solution paths with estimates $\mathbf{f}' = (f_1', f_2', \dots, f_q')$ such that,

$$\forall i f_i' \leq f_i^*$$

Therefore, the number of partial solution paths preferable equal or indifferent to \mathbf{f}^* is also necessarily finite, and P^* will be selected after all those paths in a finite number of steps. \square

Note that preservation of \leq is not enough to guarantee cautious pruning. Some accumulative functions, like maximum (max) do not preserve $<$, and therefore cannot be used safely with PRIMO- Z^* .

7.3.2. Procedure PRIMO- Z_L^*

This procedure has more relaxed conditions for admissibility than PRIMO- Z^* , though pruning may not be as effective. The pruning preference is the Pareto order with equality tie-breaking.

Proposition 11. PRIMO- Z_L^* is 1-admissible for finite graphs if,

- (a) $H(n)$ is admissible.
- (b) All attribute functions preserve equality and one of $<$ or \leq .

It is also 1-admissible for infinite graphs if additionally,

- (c) Estimate vectors in $F(P)$ are not bounded in any of their components for infinite paths.

Proof. Let us check that the admissibility conditions of POP* are satisfied:

- The proofs for conditions 1, 2, 3 and 5 are identical to those presented for the PRIMO- Z^* algorithm (see Section 7.3.1).
- Let us now consider condition 4. Lexicographic optima are, by definition, among Pareto optima. Since preservation of equality and one of $<$ or \leq is 1-cautious with Pareto optima, it is also 1-cautious with lexicographic optima. \square

7.4. Goal satisfaction search

Although goal satisfaction is a solid decision-making paradigm (Romero, 1991) with many recognized practical applications, it has not received much attention in graph search contexts. METAL- A^* , the first heuristic search algorithm developed for goal satisfaction (Fernández et al., 1999), is an instance of the procedure described here. In fact, it was largely developed in parallel with the work described in this paper.

Goal satisfaction preferences can be stated as a set of goals over solution attributes. Each goal results from the combination of an attribute $g_i(x)$ with a non-negative target value t_i , using inequality relations, e.g.,

$$\forall i g_i(x) \leq t_i$$

Goals may have associated weights w_i , and/or be organized into preemptive priority levels (the so-called lexicographic goal satisfaction). These preferences can be formally described as follows:

- Weighted goals:

$$\forall \mathbf{g}, \mathbf{g}' \in \mathbb{R}^q, \quad \mathbf{g} \prec_g \mathbf{g}' \iff p(\mathbf{g}) < p(\mathbf{g}') \vee (p(\mathbf{g}) = p(\mathbf{g}') \wedge \forall i g_i \leq g'_i \wedge \mathbf{g} \neq \mathbf{g}')$$

where the following assumptions are made: all goals take the form $g_i \leq t_i$; all attributes are of the kind “the less the better”; and $p(\mathbf{g})$ is a measure of the weighted deviation of \mathbf{g} from the set of targets.

- Lexicographic goals:

$$\forall \mathbf{g}, \mathbf{g}' \in \mathbb{R}^q, \quad \mathbf{g} \prec_g \mathbf{g}' \iff (\exists j/p_j(\mathbf{g}) < p_j(\mathbf{g}') \wedge \forall i < j p_i(\mathbf{g}) = p_i(\mathbf{g}')) \vee (\forall k p_k(\mathbf{g}) = p_k(\mathbf{g}') \wedge \forall i g_i \leq g'_i \wedge \mathbf{g} \neq \mathbf{g}')$$

where the following assumptions are made: goals are grouped in preemptive priority levels; goals in priority level i take the form $g_{ij} \leq t_{ij}$; all attributes are of the kind “the less the better”; and $p_i(\mathbf{g})$ is a measure of the weighted deviation of \mathbf{g} from the set of targets of all goals in level i .

There are several proposals for deviation functions $p(\mathbf{g})$ in the literature. The usual ones are (Romero, 1991),

- Weighted sum of deviations:

$$p(\mathbf{g}) = \sum_{i=1}^q w_i \times \max\{0, g_i - t_i\}$$

- Maximum weighted deviation (minmax strategy):

$$p(\mathbf{g}) = \max_i \{w_i \times \max\{0, g_i - t_i\}\}$$

These can additionally be normalized dividing each term by the corresponding target t_i , provided that $\forall i \ t_i \neq 0$. The algorithm described below works very well with all of these kinds of deviation measures.

7.4.1. Procedure METAL- Z_L^*

Using a goal-satisfaction preference for pruning does not seem to lead to an admissible algorithm in any case. Just consider that deviation measures are, in fact, multiattribute functions, and that none of them are among the ones described in Section 7.2.1 for Z^* . The only alternative analyzed here for pruning is the Pareto order with equality tie-breaking. The resulting algorithm is called METAL- Z_L^* (METAs Lexicograficas Z^* —Spanish translation for ‘lexicographic goals Z^* ’).

Proposition 12. METAL- Z_L^* is 1-admissible for finite graphs if,

- (a) $H(n)$ is admissible.
- (b) All attribute functions preserve equality and \leq .

It is also 1-admissible for infinite graphs if additionally,

- (c) Estimate vectors in $F(P)$ are not bounded in any of their components for infinite paths.

Proof. Let us check that the admissibility conditions of POP* are satisfied:

- Admissibility of $H(n)$ implies conditions 1 and 2 (see Proposition 6).
- The (lexicographic) goal preferences are trivially transitive (condition 3).

- Let us examine condition 4. Lexicographic goal optima and weighted goal optima are among Pareto optima. Since preservation of equality and \leq is 1-cautious with Pareto optima, it will be so with (lexicographic) goal optima.
- The proof of condition 5 is as follows. Let $\mathbf{f}^* = (f_1^*, f_2^*, \dots, f_q^*)$ be the heuristic estimate of any reachable minimal solution path P^* . Condition (c) implies that there can only be a finite number of partial solution paths with estimates $\mathbf{f}' = (f'_1, f'_2, \dots, f'_q)$ such that,

$$\forall i \ f'_i \leq f_i^*$$

Therefore, the number of partial solution paths preferable, equal or indifferent to \mathbf{f}^* is also necessarily finite, and P^* will be selected after all those paths in a finite number of steps. \square

8. An N -admissible version of POP*

Sometimes it is interesting to find the set of all minimal solutions to a multicriteria problem. This is normally the case when multiobjective preferences are used, and can also be occasionally for the other usual decision rules.

It is possible to extend POP*, POP- Z^* , and all the procedures described in previous sections, to be N -admissible. These new procedures will receive the same names, but the superscript $*$ will be replaced by N* to distinguish them from their 1-admissible counterparts.

This section describes the necessary extensions needed to turn POP* into POP N* . Then POP- Z^{N*} and its family of procedures can be easily developed with reasoning analogous to that presented in previous sections. The pseudocode for POP N* is shown in Table 5.

POP* and POP N* are quite similar. However, some important differences stem from their different termination conditions. Search in POP N* does not stop when the first solution is found. Instead, search always continues until all alternatives have been exhausted, and no more minimal solutions can be found (step 2).

As new (minimal) solutions are found, they need to be recorded (step 4.1.1) until termination, when they are all finally returned. POP N* assumes

Table 5
Pseudocode of POP^{N*}

1. Initialize-memory(*s*)
2. If not (open-paths-in-memory(*s*)), then return (recover-solutions-from-memory(*s*))
3. $P \leftarrow$ minimal-open-path-in-memory(*s*)
4. If destination-node(last-node(*P*))
 - then
 - 4.1.1. record-new-solution-in-memory(*P*)
 - else
 - 4.2.1. For each node $n \in \text{SCS}(\text{last-node}(P))$
 - Extend-paths-in-memory(last-node(*P*), *n*)
 - 4.2.2. Close-associated-paths(last-node(*P*))
5. Filter-open-paths-in-memory(*s*)
6. Return to step 2

solution paths can be recorded in the memory structure. This can be as simple as adding a new ‘solution’ label to paths in memory. Path extension (steps 4.2.1 and 4.2.2) proceeds as in POP* for non-solution paths.

Minimal solutions are an important source of information to guide search. If a minimal solution path *P* is known, then all open partial solution paths dominated by *P* can be discarded. This operation is a new special kind of pruning, called filtering, that is carried out in step 5. Filtering is not only important to reduce search effort, it also plays a vital role in guaranteeing termination on infinite graphs, as explained below.

8.1. Memory operations

In addition to the memory operations described in Section 3.1 for POP*, the procedure POP^{N*} uses the following ones:

(a) Constructors (operations that change the memory structure):

- Record-new-solution-in-memory(*P*): The path *P*, that must be already present in memory, is given a new extra ‘solution’ label, and the label ‘open’ is changed to ‘closed’.
- Filter-open-paths-in-memory(*s*): Let HEV be the set of heuristic evaluation vectors of all paths currently labeled ‘solution’ in memory. For each open path such that all its evaluation vectors are dominated by vectors in HEV according to \prec_f , its label is changed from ‘open’ to ‘closed’. It is op-

tional to remove these closed paths from memory. Removing them saves space, but keeping them may provide future chances for pruning.

(b) Selectors (operations that do not change the memory structure):

- Recover-solutions-from-memory(*s*): All paths in memory labeled as solutions are grouped and returned in a set.

8.2. Example

Let us consider a new example, similar to the one presented in Section 2.1.

- *G* is the graph in Fig. 3, all arcs are bi-directional, $s = n_1$, and $\Gamma = \{n_5, n_6\}$.
- *Preference*: The same attributes described in Section 2.1 will be used in this example. However, this time a goal-based preference will be defined as follows:

$$g_1(P) \leq 5 \text{ km}, \quad g_2(P) \geq 0.9$$

Recall that the first attribute is of the kind “the less the better”, while the second one is of the kind “the more the better”. The following deviation function will be used (normalized weighted deviation),

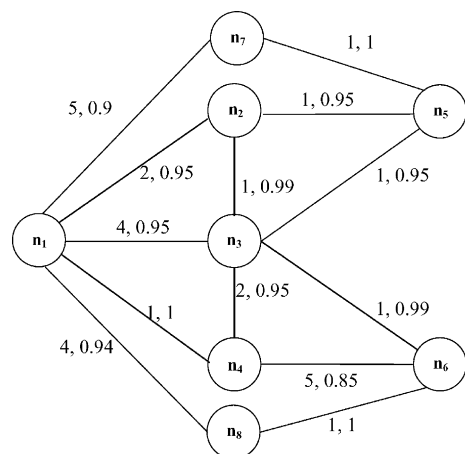


Fig. 3. A sample graph. Arcs are labeled with two values: (1) length (in kilometers); (2) probability that hungry bears will not be found.

$$p(\mathbf{g}) = \frac{\max\{0, g_1 - 5\}}{5} + \frac{\max\{0, 0.9 - g_2\}}{0.9}$$

- Heuristic preference. A single heuristic evaluation vector will be calculated for each path,

$$\mathbf{f}(P) = \mathbf{g}(P) \circ \mathbf{h}(\text{last-node}(P))$$

Heuristic estimates $\mathbf{h}(n)$ are provided in Table 6. The heuristic preference will be the goal satisfaction preference described above, but applied to heuristic estimate vectors.

- *Pruning*: The attribute vectors will also be used as pruning vectors, i.e., $\mathbf{e}(P) = \mathbf{g}(P)$. The pruning preference will be the Pareto order applied to paths reaching the same node.

Table 7 summarizes a trace of POP^{N*} applied to this problem. Relevant information is provided for each iteration. Initially only path (n_1) is in memory. At iteration 1, it is selected and five new paths are stored in memory. Their attribute and heuristic estimate vectors are shown. In order to select a minimal alternative for expansion, the deviation of heuristic evaluation vectors from the goals, $p(\mathbf{f})$, is calculated. Four of them satisfy the goals, but only one is minimal according to the heuristic preference. As search proceeds through iterations 2, 3, and 4, several new paths are pruned. At iterations 3 and 4 there are several open paths with minimal heuristic evaluation vectors, so ties are broken arbitrarily.

At iteration 5 a minimal solution path is selected, (n_1, n_2, n_5). Its heuristic evaluation vector (3, 0.9025) is added to the set HEV (and shown underlined in Table 7). This allows filtering two open paths from memory. Obviously, since a solution has been found with $p(\mathbf{f}) = 0$, no candidate

Table 6
Heuristic estimates to goal nodes n_5 and n_6 (see graph in Fig. 3)

Node	$h(n)$ (kilometers, probability of no bears)
n_1	(3, 1)
n_2	(1, 1)
n_3	(1, 1)
n_4	(2, 1)
n_5	(0, 1)
n_6	(0, 1)
n_7	(1, 1)
n_8	(1, 1)

with a worse heuristic estimate will be allowed to be open any more.

Again, two open paths qualify for expansion. The tie is broken arbitrarily in favor of the minimal solution (n_1, n_3, n_6). This allows filtering path (n_1, n_8). Finally, the only remaining minimal solution path (n_1, n_2, n_3, n_5) is selected. There are no more open paths in memory, so POP^{N*} terminates returning the set of all recorded minimal solutions,

(n_1, n_2, n_5) with attribute vector (3, 0.9025)

(n_1, n_3, n_6) with attribute vector (5, 0.9405)

(n_1, n_2, n_3, n_6) with attribute vector (4, 0.9311)

8.3. Properties of POP^{N*}

N -admissibility is a desirable property for POP^{N*}. Fortunately it can be easily proved under the same conditions used for POP* with one obvious exception: the pruning preference needs to be N -cautious according to (\mathbf{g}, \prec_g) . In other words, all minimal solution paths need to be reachable.

Lemma 3. *If (E, \prec_e) is N -cautious according to (\mathbf{g}, \prec_g) , (F, \prec_f) is optimistic, \prec_f transitive, and at least one solution exists, then, for each minimal solution P^* , there is always an open path from PSSP(P^*) in memory before P^* is selected for expansion.*

Proof. Since (E, \prec_e) is N -cautious, all minimal solution paths are reachable by definition.

The same reasoning presented for Lemma 1 (see Section 4) suffices to prove that all open paths from PSSP(P^*) cannot be pruned before P^* is finally found. However, now it is also necessary to prove that they will not be filtered either. To prove this let P' be any minimal solution path such that $F(P') = \{\mathbf{f}'\}$ and $P_{0i} \in \text{PSSP}(P')$ is open in memory (this is at least initially true for $P_{00} = (s)$). Since (F, \prec_f) is optimistic,

$$\exists \mathbf{f}'' \in F(P_{0i}) / \mathbf{f}'' \preceq_f \mathbf{f}' \tag{1}$$

If no solution has been found yet, then the filtering operation does not change the contents of memory. Let us assume some minimal solutions have already been found, and that HEV is the set of heuristic evaluation vectors of all paths

Table 7
Trace of POP^{N*}

It.	Path selected	Paths considered (● = closed, ! = solution)	$g(P)$	$f(P)$ for open paths	$p(f)$ for open paths	Pruned? (p.) filtered? (f.)	Minimal f open paths (x)
0	–	n_1	(0, 1)	(2, 1)	0		x
1	n_1	n_1 ●	(0, 1)	–	–		–
		n_1-n_2	(2, 0.95)	(3, 0.95)	0		–
		n_1-n_3	(4, 0.95)	(5, 0.95)	0		–
		n_1-n_4	(1, 1)	(3, 1)	0		x
		n_1-n_7	(5, 0.90)	(6, 0.90)	0.20		–
		n_1-n_8	(4, 0.94)	(5, 0.94)	0		–
2	n_1-n_4	n_1 ●	(0, 1)	–	–		–
		n_1-n_2	(2, 0.95)	(3, 0.95)	0		x
		n_1-n_3	(4, 0.95)	(5, 0.95)	0		–
		n_1-n_4 ●	(1, 1)	–	–		–
		n_1-n_7	(5, 0.90)	(6, 0.90)	0.20		–
		n_1-n_8	(4, 0.94)	(5, 0.94)	0		–
		$n_1-n_4-n_3$	(3, 0.95)	(5, 0.95)	0	p. (by n_1-n_3)	–
	$n_1-n_4-n_6$	(6, 0.85)	(6, 0.85)	0.25		–	
3	n_1-n_2	n_1 ●	(0, 1)	–	–		–
		n_1-n_2 ●	(2, 0.95)	–	–		–
		n_1-n_3	(4, 0.95)	(5, 0.95)	0		x
		n_1-n_4 ●	(1, 1)	–	–		–
		n_1-n_7	(5, 0.90)	(6, 0.90)	0.20		–
		n_1-n_8	(4, 0.94)	(5, 0.94)	0		–
		$n_1-n_4-n_6$	(6, 0.85)	(6, 0.85)	0.2555		–
		$n_1-n_2-n_5$	(3, 0.9025)	(3, 0.9025)	0		x
		$n_1-n_2-n_3$	(3, 0.9405)	(4, 0.9405)	0		x
4	n_1-n_3	n_1 ●	(0, 1)	–	–		–
		n_1-n_2 ●	(2, 0.95)	–	–		–
		n_1-n_3 ●	(4, 0.95)	–	–		–
		n_1-n_4 ●	(1, 1)	–	–		–
		n_1-n_7	(5, 0.90)	(6, 0.90)	0.20		–
		n_1-n_8	(4, 0.94)	(5, 0.94)	0		–
		$n_1-n_4-n_6$	(6, 0.85)	(6, 0.85)	0.2555		–
		$n_1-n_2-n_5$	(3, 0.9025)	(3, 0.9025)	0		x
		$n_1-n_2-n_3$ ●	(3, 0.9405)	–	–		–
		$n_1-n_3-n_2$	(5, 0.9405)	–	–	p. (by n_1-n_2)	–
		$n_1-n_3-n_5$	(5, 0.9025)	–	–	p. (by $n_1-n_2-n_5$)	–
		$n_1-n_3-n_6$	(5, 0.9405)	(5, 0.9405)	0		x
		$n_1-n_3-n_4$	(6, 0.9025)	–	–	p. (by n_1-n_4)	–
		$n_1-n_2-n_3-n_5$	(4, 0.8935)	–	–	p. (by $n_1-n_2-n_5$)	–
		$n_1-n_2-n_3-n_6$	(4, 0.9311)	(4, 0.9311)	0		x
$n_1-n_2-n_3-n_4$	(5, 0.8935)	–	–	p. (by n_1-n_4)	–		
5	$n_1-n_2-n_5$	n_1 ●	(0, 1)	–	–		–
		n_1-n_2 ●	(2, 0.95)	–	–		–
		n_1-n_3 ●	(4, 0.95)	–	–		–
		n_1-n_4 ●	(1, 1)	–	–		–
		n_1-n_7	(5, 0.90)	(6, 0.90)	0.20	f. (by $n_1-n_2-n_5$)	–
		n_1-n_8	(4, 0.94)	(5, 0.94)	0		–
		$n_1-n_4-n_6$	(6, 0.85)	(6, 0.85)	0.2555	f. (by $n_1-n_2-n_5$)	–
		$n_1-n_2-n_5$ ● !	(3, 0.9025)	(3, 0.9025)	–		–

Table 7 (continued)

It.	Path selected	Paths considered (● = closed, ! = solution)	$g(P)$	$f(P)$ for open paths	$p(f)$ for open paths	Pruned? (p.) filtered? (f.)	Minimal f open paths (x)
		$n_1-n_2-n_3$ ●	(3, 0.9405)	–	–		–
		$n_1-n_3-n_6$	(5, 0.9405)	(5, 0.9405)	0		x
		$n_1-n_2-n_3-n_6$	(4, 0.9311)	(4, 0.9311)	0		x
6	$n_1-n_3-n_6$	n_1 ●	(0, 1)	–	–		–
		n_1-n_2 ●	(2, 0.95)	–	–		–
		n_1-n_3 ●	(4, 0.95)	–	–		–
		n_1-n_4 ●	(1, 1)	–	–		–
		n_1-n_8	(4, 0.94)	(5, 0.94)	0	f. (by $n_1-n_3-n_6$)	–
		$n_1-n_2-n_5$ ● !	(3, 0.9025)	(3, 0.9025)	–		–
		$n_1-n_2-n_3$ ●	(3, 0.9405)	–	–		–
		$n_1-n_3-n_6$ ● !	(5, 0.9405)	(5, 0.9405)	–		–
		$n_1-n_2-n_3-n_6$	(4, 0.9311)	(4, 0.9311)	0		x
7	$n_1-n_2-n_3-n_6$	n_1 ●	(0, 1)	–	–		–
		n_1-n_2 ●	(2, 0.95)	–	–		–
		n_1-n_3 ●	(4, 0.95)	–	–		–
		n_1-n_4 ●	(1, 1)	–	–		–
		$n_1-n_2-n_5$ ● !	(3, 0.9025)	(3, 0.9025)	–		–
		$n_1-n_2-n_3$ ●	(3, 0.9405)	–	–		–
		$n_1-n_3-n_6$ ● !	(5, 0.9405)	(5, 0.9405)	–		–
		$n_1-n_2-n_3-n_6$ ● !	(4, 0.9311)	(4, 0.9311)	–		–

currently labeled ‘solution’. Let us assume then that P_{0i} is filtered, i.e.,

$$\forall \mathbf{f}'_{0i} \in F(P_{0i}) \quad \exists \mathbf{f}^* \in \text{HEV}/\mathbf{f}^* \prec_f \mathbf{f}'_{0i} \quad (2)$$

However, applying the transitive property of \prec_f to the results of (1) and (2) results in a contradiction that P' is minimal,

$$\begin{aligned} \exists \mathbf{f}^* \in \text{HEV}, \quad \exists \mathbf{f}'' \in F(P_{0i})/\mathbf{f}^* \prec_f \mathbf{f}'' \preceq_f \mathbf{f}' \\ \Rightarrow \mathbf{f}^* \preceq_f \mathbf{f}' \quad \square \end{aligned}$$

Proposition 13. *POP^{N*} always terminates on finite graphs.*

Proof. Identical to the proof of Proposition 1 (see Section 4). \square

Proposition 14. *POP^{N*} is N-admissible on finite graphs under the following (sufficient) conditions,*

1. *The heuristic preference structure (F, \prec_f) is optimistic.*

2. *The heuristic preference structure (F, \prec_f) agrees with (\mathbf{g}, \prec_g) .*

3. *The heuristic preference \prec_f is transitive.*

4. *The pruning preference (E, \prec_e) is N-cautious according to (\mathbf{g}, \prec_g) .*

Proof. The proposition states that POP^{N*} terminates with the set of all minimal solutions according to (\mathbf{g}, \prec_g) whenever at least one exists and the set is finite. However, by condition 2 this amounts to proving that POP^{N*} terminates with all minimal solutions according to (F, \prec_f) whenever at least one exists and the set is finite.

Since G is finite, then the sets of all partial solutions and of all minimal solutions is necessarily finite. From Proposition 13 the procedure always terminates on finite graphs. The only termination condition is that all paths in memory are closed. Therefore, all minimal solution paths will be necessarily found unless some subpath of them is either pruned or filtered previously. However, Lemma 3 proves that this cannot be the case. Consequently, all minimal solutions will be finally selected and labeled as solutions.

Let us prove now that dominated (non-minimal) solutions will never be selected and labeled as solutions. Let P' be a dominated solution. There exists by definition a minimal solution P^* such that $F(P^*) = \{\mathbf{f}^*\}$, $F(P') = \{\mathbf{f}'\}$, and $\mathbf{f}^* \prec_f \mathbf{f}'$.

If P^* has not been found yet, then by lemma 3 there exists some open path $P_{0i} \in \text{PSSP}(P^*)$ in memory such that,

$$\exists \mathbf{f}_{0i} \in F(P_{0i}) \wedge \mathbf{f}_{0i} \preceq_f \mathbf{f}^* \prec_f \mathbf{f}'$$

Consequently, P will never be selected prior to P^* .

If P^* has already been found, then $\mathbf{f}^* \in \text{HEV}$. Therefore, if P is ever included and open in memory, it will be immediately filtered. \square

Proposition 15. *POP^{N*} is N-admissible on infinite graphs if, in addition of the conditions presented in Proposition 14, the following condition holds,*

5. *For each minimal solution path P^* such that $F(P^*) = \{\mathbf{f}^*\}$, there is only a finite number of partial solution paths P' such that,*

$$\exists \mathbf{f}' \in F(P') / (\mathbf{f}' \preceq_f \mathbf{f}^* \vee \mathbf{f}' = \mathbf{f}^*)$$

Proof. Note that condition 5 implies (by definition) that the set of minimal solution paths is finite. Each minimal solution will be found in a finite number of steps after expansion of all partial solution paths with preferable, indifferent or equal estimates. Since the set of minimal solutions is finite, it will be found in a finite number of iterations.

Condition 5 also implies that the set of all partial solution paths not dominated by some minimal solution is also finite. Therefore, after all minimal solutions are found, all paths in memory will be eventually closed, pruned or filtered in a finite number of steps. \square

8.4. Instances of POP^{N*}

Proposition 16. *If (E, \prec_e) preserves (\mathbf{g}, \prec_g) , then it is N-cautious.*

Proof. Trivial from the results presented in Section 5. \square

This result opens the way to POP-Z^{N*} and its family of procedures. It is unnecessary to repeat all the discussion presented in previous sections. Obviously N-admissible procedures do not use equality tie-breaking in pruning.

9. Selecting the right algorithm

This section is intended to serve as a guide to the selection of the right algorithm for a particular multicriteria search problem. Therefore, it summarizes in practical terms the results presented in previous sections. Several implementation details are also discussed at the end of this section. The following assumptions apply,

- (1) A graph search problem (G, s, Γ) and the DMs preference structure (\mathbf{g}, \prec_g) have already been formally defined (see Section 2).
- (2) The requested solution is either a minimal solution or, more likely, the set of all minimal solutions.

The minimality requirement forces the selection of admissible algorithms. If this requirement were relaxed, a different set of techniques might also be appropriate.

The selection of an admissible algorithm is divided into two steps. These consider in turn which heuristic (F, \prec_f) and pruning preferences (E, \prec_e) are suitable for a given DMs preference (\mathbf{g}, \prec_g) . Fig. 4 summarizes the decision process described below for N-admissible algorithms. However, the same applies to 1-admissible ones.

9.1. Decision maker's preference

The selection process outlined in Fig. 4 depends completely on the DMs preference (\mathbf{g}, \prec_g) . This preference is defined by an attribute function over paths $\mathbf{g}(P)$, and some preference relation between attribute vectors \prec_g . While many different kinds of attributes and preference relations are possible from a mathematical point of view, special care has been taken in previous sections in order to present a small but highly representative set of possibilities:

- Only four typical preference relations have been discussed in this paper: multiobjective, multiattribute, goal satisfaction, and lexicographic. Their formal definitions can be found in Section 7.
- In the case of attribute functions, an important distinction has been made between cumulative

and non-cumulative ones. Basically, cumulative attributes use some associative combining operator over some property of the arcs/nodes of a path (see Section 6). Some examples of cumulative combining operators are: addition (+); multiplication (\times); obtaining the maximum of a set (max); or obtaining the minimum of a set (min).

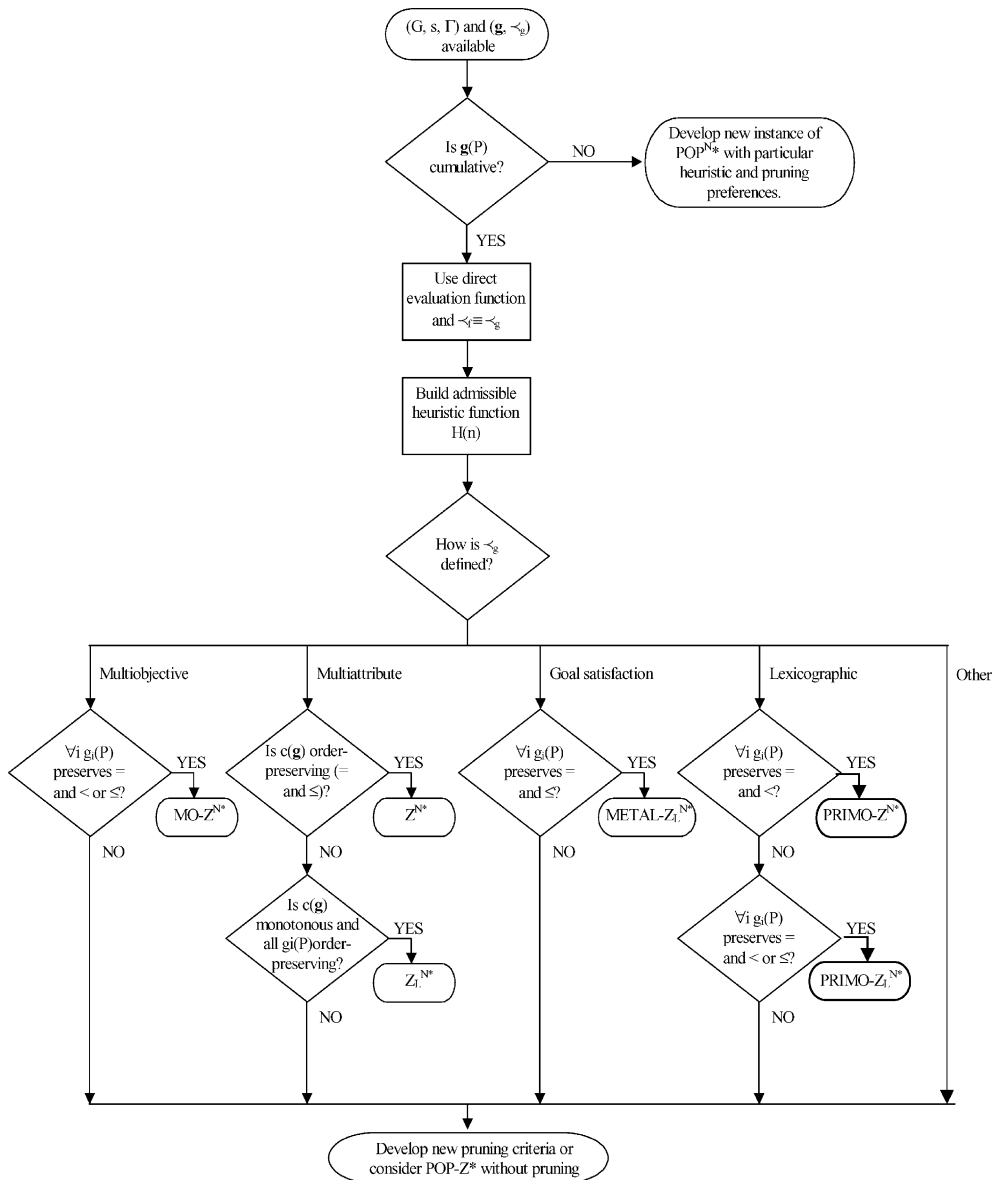


Fig. 4. Selection of heuristic and pruning preferences. Search involves a graph G that is not a tree.

Addition is the most frequent combining operator, e.g., it can be used to add costs or distances associated to arcs in a path. Multiplication is frequently associated to probability calculations. The max and min operators are useful in ‘minmax’ or ‘maxmin’ decision strategies, e.g., ‘minimizing maximum arc costs in a path’, or ‘maximizing minimum arc capacities in a path’. An example of a non-cumulative combining function could be the median of a set. The median of a set of numbers N is the $(|N|/2)$ th smallest number in N .

The selection of the right algorithm for a given DMs preference depends both on the cumulative/non-cumulative nature of the attributes, and on the kind of preference relation under consideration.

9.2. Heuristic preference

The first step in the selection process defines a suitable heuristic preference (F, \prec_f) . In order to guarantee admissibility, both the heuristic and DMs preference must have the same set of minima when applied to a given set of alternatives. In such a case both preferences are said to agree.

The nature of the attributes, $\mathbf{g}(P)$, largely determines future decisions. Practical applications of best-first search algorithms involve cumulative attributes, and most frequently, additive ones. Otherwise, it is not clear how a heuristic preference should be constructed, or what kind of pruning preference should be used. When non-cumulative attributes are involved, new instances of POP^{N*} need to be devised.

However, when the DMs preference involves only cumulative attributes, the heuristic preference can always take the following form:

- (a) The heuristic preference relation \prec_f is the same as the DMs \prec_g .
- (b) F is a direct evaluation function (see Definition 11, Section 6.2).

In this case, the only additional element needed to construct the heuristic preference is an admissible multicriteria heuristic function $H(n)$ (see

Definitions 10 and 12, Sections 6.2 and 6.3). Pearl (1984, Chapter 4) discusses how admissible heuristic functions can be devised for individual attribute functions. Good heuristic functions can reduce search effort considerably in certain problem instances. However, obtaining them is a difficult task for most attributes, and strongly depends on the particular features of the problem being solved. An important exception are additive attributes related to some kind of physical distance. In this case well-known heuristics are available (see Pearl, 1984, pp. 140–149).

A simple admissible multicriteria heuristic function could be,

$$H(n) = \{(h_1(n), h_2(n), \dots, h_q(n))\}$$

where each $h_i(n)$ is an admissible heuristic function for the scalar $g_i(P)$ attribute in the sense described by Pearl (1984). The following are admissible heuristic functions for different kinds of scalar attributes:

Kind of attribute	Attribute values	Combining operator	Heuristic function
The less the better	Non-negative real	+	$\forall n \ h(n) = 0$
The more the better	$[0, 1]$	\times	$\forall n \ h(n) = 1$
The less the better	$[k_1, k_2]$	max	$\forall n \ h(n) = k_1$

While the heuristic functions presented above do not add new information to cut down the search effort (i.e., they amount to what is usually called blind or uninformed search), at least they allow the use of these attributes in (\mathbf{g}, \prec_g) without compromising admissibility.

9.3. Pruning preference

The second step in the selection process involves choosing a suitable pruning preference. The pruning preferences discussed in this paper apply only to paths reaching the same node. Therefore, this decision is relevant only when the graph G

being searched is not a tree. The power of pruning can reduce the search effort exponentially in certain problem instances and should not be underestimated (see, for example, Russell and Norvig, 1995, Section 3.6).

In order to guarantee admissibility, the minima of the DMs preference (\mathbf{g}, \prec_g) must be among the minima of the pruning preference (E, \prec_e) or (\mathbf{e}, \prec_e). In such a case, the pruning preference is said to be N -cautious according to the DMs preference. Whereas the work of Carraway et al. (1990) opens the possibility of heuristic pruning, this paper only analyzes the case where $\mathbf{g}(P) = \mathbf{e}(P)$.

Two different alternatives for the preference relation \prec_e are discussed in Section 7: using Pareto order, or using the DMs preference relation \prec_g . These do not always guarantee admissibility, and careful analysis must be made depending on \prec_g . The results of this analysis are presented in Section 7, and are briefly summarized in Fig. 4.

9.4. Implementation details

Some implementation details of the algorithms were intentionally omitted in the previous discussion for the sake of clarity. These involve the structure and management of the memory and are briefly discussed here from a practical point of view. Two related issues need to be considered in this sense:

- (a) The effective storage of the generated partial solution paths.
- (b) The selection of the minimal path at each iteration.

The most usual structure to store partial solution paths is a labeled acyclic graph. More details on the use of acyclic graphs in multicriteria search algorithms can be found in (Stewart and White, 1991; Fernández et al., 1999).

In addition to a graph/tree with all partial solution paths, most shortest path algorithms claim to maintain a list of ‘open’ paths. Two categories are usually found in the shortest path literature, those of ‘label setting’ and ‘label correcting’ algo-

rithms. The former select a minimal alternative at each iteration. Therefore, they need efficient data structures to keep them properly sorted after each new addition or deletion from the set. On the other hand, label correcting algorithms do not follow this selection policy and alternatives are added and deleted from a queue using simple constant time operations. This results in a much faster selection process at the expense of increasing the number of iterations. The algorithms described in this paper fall within the label setting class.

Traditionally, label setting algorithms have been considered better than label correcting ones when it comes to computing the shortest path between a source node and a single destination node in a graph. However, it is important to note that recent results have begun to challenge this view in certain situations (Zahn and Noon, 2000).

In order to reduce the overhead of alternative selection at each iteration, label setting algorithms use a *priority queue* that can be implemented in several ways (e.g. as a heap) (Knuth, 1973; Carlsson and Chen, 1992). Sorting algorithms for these data structures use some simple total order relation (typically $<$) to sort elements according to their scalar cost values. Multicriteria search algorithms also need efficient means to sort candidate partial solution paths. However, in this case, paths need to be sorted according to a heuristic attribute vector and a multicriteria preference relation. Different possibilities arise, depending on the nature of this preference relation,

- Since lexicographic and multiattribute preferences induce a total order on attribute vectors, they can be used directly as an order relation to sort alternatives in a priority queue. The first element in the queue will always be minimal according to the lexicographic/multiattribute preference.
- In the multiobjective case, paths should be sorted in the queue in such a way that the first element in the queue is always Pareto optimal. Regrettably, Pareto optimality induces only a partial order. However, any arbitrary lexicographic order can be used here to sort ‘open’ alternatives, since lexicographic minima are also Pareto optima. Therefore, a Pareto optimal

alternative will always be the first element in the queue.

- Goal satisfaction preferences are not enough to sort the priority queue either. In this case, a minimal alternative can be placed at the top of the queue sorting alternatives lexicographically according to the prioritized deviation measures (to guarantee one of the most satisfactory alternatives at the top of the heap), and breaking ties with an arbitrary lexicographic order of the attributes (to guarantee that the top of the queue is also Pareto optimal).

10. Related work

Fig. 2 displays the relationship between POP*, POP-Z*, and their family of related procedures described in previous sections. An analogous graphic could be used to describe their N -admissible counterparts. This framework, and the conceptualization behind POP*, is a useful tool to describe the achievements of previous work in the field of multicriteria graph search. Algorithms previously described in the literature are shown inside squares in Fig. 2.

As it turns out, most previous contributions are related to the multiattribute paradigm, perhaps because it is conceptually closer to scalar optimization. The Z^* algorithm (Pearl, 1984) can be very easily adapted to multiattribute search. The only necessary change is to keep an accumulative attribute vector associated with each partial solution path, instead of a single scalar attribute. The celebrated algorithm A^* is an instance of Z^* , and can also be adapted to multicriteria search in a similar way.

Regrettably, this simple adaptation of Z^* does not lead to an admissible multicriteria algorithm except for the two cases described in Section 7.2.1. This led (Loui, 1983) to propose the Pareto order as (local) pruning preference in his multiattribute generalization of Dijkstra's algorithm (the uninformed counterpart of A^*). This paper proposes a heuristic version of Loui's algorithm called Z_L^* . While Pareto preference is likely to prune fewer paths than multiattribute preference, it results in admissible search for a more general class of multiattribute functions.

The quest for general and more effective pruning preferences in multiattribute dynamic programming resulted in the interesting work by Carraway et al. (1990). These authors describe a cautiousness condition different from the one proposed in Section 5, that results in an admissible search (GDP) that is more effective than Pareto pruning. An important disadvantage of GDP is the need of additional information that is normally not available, but that can nevertheless be estimated heuristically. This opens the attractive possibility of heuristic pruning. The work of Mandow (1999) proves that the conditions of GDP imply N -cautiousness.

Unfortunately, the interesting ideas behind GDP have been used so far only in the limited multiattribute search algorithm U^* (White et al., 1992). This algorithm uses a highly unlikely kind of heuristic estimate. Therefore, it is not an instance of POP-Z*, although it can still be described as an instance of POP* (see Mandow, 1999).

Another interesting contribution to multicriteria search is the MOA* algorithm (Stewart and White, 1991). This is an N -admissible multiobjective counterpart of A^* , that under the terminology of this paper receives the name of MO-A ^{N^*} . However, in the work of (Stewart and White, 1991) the explicit conceptualization behind POP* appears blurred, probably due to the fact that the preferences \prec_g , \prec_f , and \prec_e are the same for additive multiobjective search. An important difference between Stewart and White's MOA* and POP ^{N^*} (and, therefore, MO-A ^{N^*}) is the management of the memory structure. POP* considers each different path for extension only once, i.e., once a path is 'closed' it is not opened anymore. However, MOA* may reopen paths during its operation under certain conditions, i.e., each path may need to be examined many times. This subtle difference has an extremely important impact on efficiency, and in fact renders MOA* unsuitable for certain problem instances, since path extension is the single basic operation in best-first algorithms (see Mandow, 1999). Again, this difference stems from a different conceptualization in which known paths (and not nodes) are considered for expansion at each iteration.

Thus far, this section has examined previous work in the light of this paper. Conversely, in the

light of previous contributions, this work has extended the ideas behind Loui's work, formalizing pruning as a preference, and analyzing admissibility conditions for multicriteria decision rules. In any case, the quest for efficient pruning criteria remains an important research topic in multicriteria search.

11. Conclusions and future work

This paper extends the multicriteria decision paradigm to the heuristic search domain in a systematic way.

A first step towards this extension is a formal definition of MHS problems. It is particularly enlightening to define these problems explicitly in terms of problem, heuristic, and pruning preferences.

Two new general procedures, POP* and POP^{N*}, have been introduced for general preference-based search. These procedures perform search using a set of partial solution paths. At each iteration a path is selected and extensions are considered for inclusion in the set. Search is constrained to minimal paths according to the pruning preference, and aims at the extension of minimal paths according to the heuristic preference. Sufficient conditions that relate the three kinds of preference have been provided to guarantee admissible search with both POP* and POP^{N*}.

As it turns out, a fundamental issue for efficient multicriteria search is finding pruning preferences that behave correctly for given problem preferences. To tackle this problem the paper provides a set of operational conditions that can be used to check whether a given pruning criterion is suitable for any given kind of problem preference.

All these results provide a general framework that makes the analysis of many multicriteria search procedures easier. Particularly, the important family of search procedures for cumulative attribute problems has been analyzed. First, a simple and effective heuristic preference has been described for these kinds of problems. This is similar to the one frequently used in analogous scalar search problems. Then, the most frequent multicriteria problem preferences (i.e., multiob-

jective, multiattribute, goal satisfaction, and lexicographic preferences) have been combined with two kinds of pruning preference: the Pareto order, and the problem's own preference.

This analysis results in six new 1-admissible search procedures (extensible to be also *N*-admissible). These can be extended in turn to provide new algorithms for the additive case. Some of these resulting algorithms are described elsewhere (Mandow et al., 1998; Fernández et al., 1999).

In sum, the results presented in this paper provide a comprehensive framework that allows the reasoned development of new multicriteria search procedures, and also relates to previous research efforts. In particular, its relationship to the work described in (Pearl, 1984; Loui, 1983; Stewart and White, 1991; Carraway et al., 1990) has been noted.

The design of particular data structures and algorithms that conform to the specification of the memory operations described for POP* and POP^{N*} is a necessary and complementary part of this paper. The effective storage of paths in memory is a common problem with known scalar optimization algorithms, and solved efficiently using trees or acyclic graphs.

Finally, formal and empirical analysis on the behavior of MHS algorithms, like the ones carried out in (Pearl, 1984) for scalar heuristic search, are an important theoretical continuation of this work.

References

- Carlsson, S., Chen J., 1992. The complexity of heaps. In: Proceedings of the Third Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 27–29 January 1992, Orlando, FA. ACM/SIAM. Available from the ACM Digital Library.
- Carraway, R.L., Morin, T.L., Moskowitz, H., 1990. Generalized dynamic programming for multicriteria optimization. *European Journal of Operational Research* 44, 95–104.
- Chankong, V., Haimes, Y.Y., 1983. *Multiobjective Decision Making. Theory and Methodology*. North-Holland, New York.
- Dijkstra, E.W., 1959. A note on two problems in connection with graphs. *Numerische Mathematik* 1, 269–271.
- Fernández, J.A., González, J., Mandow, L., Pérez de la Cruz, J.L., 1999. Mobile robot path planning: A multicriteria approach. *Engineering Applications of Artificial Intelligence* 12, 543–554.

- Hart, P.E., Nilsson, N.J., Raphael, B., 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on System Science and Cybernetics* SSC-4, 100–107.
- Hart, P.E., Nilsson, N.J., Raphael, B., 1972. Correction to A formal basis for the heuristic determination of minimum cost paths'. *SIGART Newsletter* 37, 28–29.
- Knuth, D.E., 1973. *The Art of Computer Programming*. Vol. 3: Sorting and Searching. Addison-Wesley, Reading, MA.
- Loui, R.P., 1983. Optimal paths in graphs with stochastic or multidimensional weights. *Communications of the ACM* 26 (9).
- Mandow, L., 1999. *Búsqueda Heurística Multicriterio para Inteligencia Artificial en Diseño (Multicriteria Heuristic Search for Artificial Intelligence in Design)*. Ph.D. dissertation. Dpto. Lenguajes y Ciencias de la Computación, Universidad de Málaga, Spain (in Spanish).
- Mandow, A., Mandow, L., Muñoz, V.F., García Cerezo, A., 1998. Multi-objective path planning for autonomous sensor-based navigation. In: Ollero, A. (Ed.), *Intelligent Components for Vehicles*. Pergamon Press, Oxford, pp. 337–381.
- Mandow, L., Pérez de la Cruz, J.L., 2000. The role of multicriteria problem solving in design. In: Gero, J.S. (Ed.), *Artificial Intelligence in Design'00*. Kluwer Academic Publishers, pp. 23–41.
- Navinchandra, D., 1991. *Exploration and Innovation in Design. Towards a Computational Model*. Springer-Verlag, New York.
- Pearl, J., 1984. *Heuristics. Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley Publishing Company, Reading, Massachusetts.
- Romero, C., 1991. *Handbook of critical issues in goal programming*. Pergamon Press, Oxford.
- Russell, S., Norvig, P., 1995. *Artificial intelligence: A modern approach*. Prentice Hall, London.
- Stewart, B.S., White III, C.C., 1991. Multiobjective A*. *Journal of the Association for Computing Machinery* 38 (4), 775–814.
- Sykes, E.A., White III, C.C., 1991. Multiobjective intelligent computer-aided design. *IEEE Transactions on Systems, Man and Cybernetics* 21 (6), 1498–1511.
- White III, C.C., Stewart, B.S., Carraway, R., 1992. Multiobjective, preference-based search in acyclic OR-graphs. *European Journal of Operational Research* 56, 257–363.
- Yu, P., 1985. *Multiple-Criteria Decision Making*. Plenum Press, New York.
- Zahn, F.B., Noon, C.E., 2000. A comparison between label-setting and label-correcting algorithms for computing one-to-one shortest paths. *Journal of Geographic Information and Decision Analysis* 4 (2), 1–11.