A Multiobjective Frontier Search Algorithm

L. Mandow and J.L. Pérez de la Cruz
Dpto. Lenguajes y Ciencias de la Computación
Universidad de Málaga 29071 - Málaga (Spain)
{lawrence, perez}@lcc.uma.es

Abstract

The paper analyzes the extension of frontier search to the multiobjective framework. A frontier multiobjective A* search algorithm is developed, some formal properties are presented, and its performance is compared to those of other multiobjective search algorithms. The new algorithm is adequate for both monotone and non-monotone heuristics.

1 Introduction

Systematic graph search procedures are usually classified into best-first search and depth-first search strategies. The former can take advantage of the so-called principle of optimality to prune paths that could never lead to solutions better than those already recorded. Dijkstra's algorithm and A* [Hart *et al.*, 1968] are members of this class. Pruning may drastically reduce the number of paths explored in the graph but memory requirements are usually exponential with solution depth. Depth-first search presents the advantage of memory requirements linear with the depth of the solution. However in certain cases these algorithms may involve the consideration of an exponentially larger set of paths when compared to best-first algorithms. Algorithms like IDA* [Korf, 1985], and RBFS [Korf, 1993] are members of this class. Several attempts have been carried out to retain the benefits of path pruning while keeping reasonable memory requirements. Frontier search [Korf *et al.*, 2005] has been recently reported as a successful approach in this sense.

Multiobjective search is known to be much more computationally demanding than its scalar counterpart. Particularly, memory requirements are one of the practical limiting factors of multiobjective best-first search in AI applications. Applications in route planning [Alechina and Logan, 2001] and domain independent planning [Refanidis and Vlahavas, 2003] have reported queue size as a limiting factor and resorted to various schemes to limit the number of generated alternatives.

Multiobjective best-first search algorithms include two different extensions of A* to the multiobjective case, MOA* [Stewart and White, 1991], and NAMOA* [Mandow and Pérez de la Cruz, 2005]. Multiobjective depth-first algorithms include IDMOA* [Harikumar and Kumar, 1996], and MOMA*0 [Dasgupta *et al.*, 1999], the multiobjective extensions of IDA* and RBFS respectively.

This paper analyzes the extension of frontier search to the multiobjective framework. Scalar frontier search exploits the monotone property of heuristics to discard unnecessary nodes, achieving important memory savings. However, as explained in the paper, the same node deletion criteria cannot be applied in a multiobjective context. This paper presents new deletion criteria that can be applied in broader contexts while preserving the properties of conventional algorithms.

The paper is organized as follows. Section 2 describes scalar frontier search. Section 3 examines the fundamental issues in multiobjective search and introduces a frontier multiobjective A* search algorithm. Formal properties and experimental results are presented in sections 4 and 5 respectively. Finally, some conclusions and future work are outlined.

2 Scalar frontier search

The shortest path problem can be stated as follows: Let G be a locally finite labeled directed graph $G = (N, A, c)$, of $|N|$ nodes, and $|A|$ arcs (n, n') labeled with positive costs $c(n, n') \in \mathbb{R}$. Given a start node $s \in N$, and a set of goal nodes $\Gamma \subseteq N$, find the minimum cost path in G from s to a node in Γ.

Scalar best-first algorithms like A* build a search tree T rooted at the start node with the best paths found to each generated node. Nodes already expanded are kept in a list of $CLOSED$ nodes, and those that are waiting for expansion are kept in a list of $OPEN$ nodes. Keeping the $CLOSED$ list makes memory requirements exponential with solution depth. However, these nodes serve two important purposes. Firstly, whenever a new path is found to a known node, its cost is compared to that of the path in the search tree. The best path is kept and the other is pruned. This can drastically reduce the number of paths in G that need to be explicitly explored. The second important use of closed nodes is to return the solution path. When a goal node is reached, the solution path can be quickly recovered from the search tree.

A* uses a characteristic evaluation function $f(n) = g(n) + h(n)$ to rank open nodes and always selects for expansion nodes with minimum $f(n)$. Function $g(n)$ denotes the cost of the path stored in the search tree from s to n, while the heuristic function $h(n)$ estimates the cost of a solution from node n to a goal node. When $h(n)$ satisfies the so-called monotone property, i.e.

$$h(n) + c(n, n') \leq h(n') \quad \forall (n, n') \in A \tag{1}$$

then for all node n selected for expansion an optimal path to n has already been found in T. The use of monotone heuristics optimizes and simplifies path pruning: new paths found to closed nodes can be pruned straightaway, since no closed node will ever be put back into $OPEN$.

In the following discussion on frontier search we shall make the following assumptions:

1. We are only interested in the cost of the optimal solution, or in which goal node is reached by an optimal solution, but not in the optimal solution path itself.
2. The graph G to be searched is undirected, i.e. for each arc (n, n') in G, there is also an arc (n', n).
3. Heuristic functions are monotone.

Under these assumptions frontier search keeps the benefits of path pruning while drastically reducing memory requirements. The key idea is to store in memory only the $OPEN$ list, but not the $CLOSED$ list of expanded nodes.

Frontier search keeps a vector of *used operators* with each node n. Each vector element indicates whether the neighboring node n' reached via that operator is still accessible or not through node n. Initially, all vector elements are set to 'unused'. Each time a node is expanded, only successors reachable by unused operators are generated. Recall that whenever a node n is selected for expansion, an optimal path to that node has been found. This means no new interesting paths to n will ever be generated and n will never again need to be put back into $OPEN$. Therefore, for each generated neighbouring node n', the operator that reaches n form n' can be marked as used. This amounts to pruning all new paths that might reach n from n' even before they are actually generated. Since n will never again be reached by new paths, it can be safely removed from memory.

Assumptions 1 and 2 presented for this basic frontier search algorithm can be relaxed to make it a more general search strategy [Korf *et al.*, 2005]. However, assumption 3 cannot be relaxed without compromising the admissibility of frontier search. With a monotone heuristic, and the same tie-breaking rule, the basic frontier search algorithm can be shown to mimic the sequence of node expansions performed by standard best-first search. Therefore, it is admissible under the same assumptions.

3 Multiobjective frontier search

3.1 Multiobjective A* search

The multiobjective search problem can be stated as follows: Let G be a locally finite labeled directed graph $G = (N, A, \vec{c})$, of $|N|$ nodes, and $|A|$ arcs (n, n') labeled with positive vectors $\vec{c}(n, n') \in \mathbb{R}^q$. Given a start node $s \in N$, and a set of goal nodes $\Gamma \subseteq N$, find the set of all *non-dominated* cost paths in G from s to nodes in Γ.

The main distinguishing feature of multiobjective problems is the fact that cost vectors $\vec{c}(n, n')$ induce only a partial order preference relation $\prec$ called *dominance*,

$$\forall \vec{f}, \vec{f'} \in \mathbb{R}^q \quad \vec{f} \prec \vec{f'} \quad \Leftrightarrow \quad \forall i \quad f_i \leq f'_i \ \wedge \ \vec{f} \neq \vec{f'} \tag{2}$$

where f_i denotes the i-th element of vector $\vec{f}$.

Therefore, given two vectors $\vec{f}$ and $\vec{f'}$, it is not always possible to rank one as better than the other. For example given vectors $\vec{x} = (1, 1)$, $\vec{y} = (3, 2)$, and $\vec{z} = (1, 3)$, $\vec{x}$ dominates both $\vec{y}$ and $\vec{z}$, but no dominance relation exists between vectors $\vec{y}$ and $\vec{z}$.

Given a set of vectors X, we shall define $nondom(X)$ the set of non-dominated vectors in set X in the following way,

$$nondom(X) = \{\vec{x} \in X \mid \nexists \vec{y} \in X \quad \vec{y} \prec \vec{x}\} \tag{3}$$

We shall use NAMOA* [Mandow and Pérez de la Cruz, 2005] as reference in the development of a frontier multiobjective A* algorithm. It is a refinement of MOA* [Stewart and White, 1991] that offers better memory behaviour. NAMOA* builds an acyclic search graph SG rooted at s to store all nondominated paths found to each node. Whenever a new path is found to a known node n, its cost is compared to those already reaching n. Nondominated paths are kept, and dominated ones are pruned. NAMOA* keeps two different sets associated to each node. $G_{op}(n)$ denotes the set of cost vectors of paths reaching n that can be further explored, while $G_{cl}(n)$ denotes the set of those that have already been expanded. The set of *$OPEN$ paths* in SG that can be further explored is made up of all tuples $(n, \vec{g})$ where $\vec{g} \in G_{op}(n)$.

NAMOA* uses a heuristic function $H(n)$ that estimates the set of nondominated cost vectors of paths from n to each goal node. Therefore, each path P_{sn} from s to n with cost $\vec{g}(P_{sn})$, has a set of heuristic evaluation vectors, $F(P_{sn})$. This function is the multiobjective analogue to $f(n)$ in A*,

$$F(P_{sn}) = nondom\{\vec{f} \mid \vec{f} = \vec{g}(P_{sn}) + \vec{h} \ \wedge \vec{h} \in H(n)\}$$

NAMOA* always selects for expansion an open path P with at least an heuristic evaluation vector f nondominated in $OPEN$. Two sets, $GOALN$ and $COSTS$, keep track of goal nodes and costs of nodominated solutions found so far. Each time a new solution is found, dominated alternatives are filtered from $OPEN$ and search proceeds until $OPEN$ is empty.

In multiobjective search many different nondominated paths may reach a given node. Therefore, it is the number of cost vectors stored in $G_{op}(n)$ and $G_{cl}(n)$ that dominates memory requirements, while the number of nodes plays only a minor role. Regrettably, the expansion of a nondominated path to node n does not prevent other nondominated paths to enter $G_{op}(n)$ at later stages of the search, even when monotone heuristics are used [Stewart and White, 1991](lemma 20). Note that even vector costs of expanded paths cannot be discarded after expansion without further consideration. They need to be kept in the $G_{cl}(n)$ sets to prune new dominated paths reaching already expanded nodes. This prevents a trivial extension of scalar frontier search to the multiobjective case. Determining whether all nondominated paths to a given node have already been found becomes then the central issue.

3.2 Frontier search NAMOA*

The extension of frontier search to multiobjective search is presented under the same assumptions presented in section 2 for its scalar counterpart. The key is to find an adequate criterion to allow nodes and cost vectors of expanded paths to be safely deleted from memory. As previously explained, this is not completely straightforward, since monotonicity does not provide the same properties as in scalar search. The algorithm presented in table 1 is based in the following proposals.

Let us denote by $G(n) = G_{op}(n) \cup G_{cl}(n)$ the set of all nondominated cost vectors known to n, and by $FRONTIER$ the set of known nodes that cannot be deleted from memory yet. Each node in $FRONTIER$ will store a vector of 'used operators', a deletion flag, and an 'at least once expanded' flag.

Candidates for deletion A node n is a candidate for deletion when no more nondominated paths can be found to n. We can be certain of this condition whenever $\forall n' \in FRONTIER \wedge \forall \vec{g}' \in G_{op}(n') \quad \exists \vec{g} \in G(n) \quad \vec{g} \preceq \vec{g}'$.

Marking used operators If node n is candidate for deletion and at least one path leading to n was expanded, then the operators of its successors leading to n can be marked as 'used'. This amounts to pruning any new paths leading to n even before they are actually generated.

Node deletion If node n is candidate for deletion and $G_{op}(n) = \emptyset$, then n can be actually deleted from memory.

These conditions are checked at each iteration at step 6. Note that new nodes generated by the algorithm must be put in the $FRONTIER$ even if they are not reached by an interesting alternative, i.e. the newly found path is dominated by $COSTS$ (step 5(b)i). These nodes would be typically discarded in MOA* or NAMOA* until a really interesting path was found to them, but need to be kept in frontier search in case some of their operators have to be marked as 'used'.

These conditions are rather severe. However, the most important condition in multiobjective frontier search involves cost vector deletion, since it is the number of cost vectors stored in memory that dominates memory consumption.

Cost vector deletion When a node n is marked for deletion, then the set $G_{cl}(n)$ can be immediately removed from memory (step 6(a)i). Additionally, each time a new path to n is selected for expansion, its cost vector can be removed from $G_{op}(n)$ and discarded (step 3b).

The rationale behind this condition is that costs in $G_{cl}(n)$ are kept to prune new dominated paths found to n. However, once a node n is marked for deletion, neighbouring nodes will mark their operators and no new path will ever reach n. Notice that, by virtue of this condition, a node marked for deletion can get rid of some its cost vectors well before the node itself is actually deleted.

3.3 Example

Figure 1(a) shows a sample graph with multiobjective costs. Let us assume s is the start node, $\Gamma = \{\gamma_1, \gamma_2\}$, and, without loss of generality, that no heuristic function is used, i.e. $H(n) = \{\vec{0}\} \ \forall n$, and the heuristic evaluation vector of each path P is $F(P) = \{\vec{g}(P)\}$. In this example we focus our attention in the criteria for path and node deletion, and these involve only real cost vectors $\vec{g}(P)$. Heuristic evaluation vectors are involved in the selection of open paths for expansion and, in this sense, FS-NAMOA* does not perform differently from NAMOA*.

Initially, s is the only node in OPEN and FRONTIER with $G_{op}(s) = \{(0,0)\}$. It is selected for expansion and paths

```
1. INITIALIZATION. Create a set of nodes $FRONTIER = \{s\}$, and set $G_{op}(s) = \{\vec{0}\}$.
   Let $OPEN$ be the list of all pairs $(n, \vec{g})$ such that $n \in FRONTIER$ and $\vec{g} \in G_{op}(n)$. Create two empty sets, $GOALN$, $COSTS$.
2. TERMINATION. If $OPEN$ is empty, then return the set of goal nodes $GOALN$ and nondominated costs $COSTS$.
3. PATH SELECTION.
   (a) Select an alternative $(n, \vec{g}_n)$ from $OPEN$ with heuristic evaluation $\vec{f} \in F(n, \vec{g}_n)$ non-dominated in $OPEN$ and not dominated by $COSTS$.
   (b) If $n$ is marked for deletion, then remove $\vec{g}_n$ from $G_{op}(n)$, else move $\vec{g}_n$ from $G_{op}(n)$ to $G_{cl}(n)$.
4. SOLUTION RECORDING. If $n \in \Gamma$, then put $n$ in $GOALN$, $\vec{g}_n$ in $COSTS$, and eliminate from $OPEN$ all dominated alternatives.
5. PATH EXPANSION: If $n \notin \Gamma$, then
   For all successors nodes $m$ of $n$ with cost $\vec{g}_m = \vec{g}_n + \vec{c}(n, m)$ do:
   (a) Calculate $F_m$ the set of estimates for the new path $F(m, \vec{g}_m)$ not dominated by $COSTS$.
   (b) If $m$ is a new node, then,
       i. Put $m$ in $FRONTIER$.
       ii. If $F_m \neq \emptyset$ then place $\vec{g}_m$ in $G_{op}(m)$.
       else, If $\vec{g}_m$ is new and nondominated in $G(m)$, then:
       i. Prune from $G(m)$ vectors dominated by $\vec{g}_m$.
       ii. If $F_m \neq \emptyset$ then place $\vec{g}_m$ in $G_{op}(m)$.
6. UPDATE FRONTIER: For all node $n$ in $FRONTIER$ do,
   (a) If $n$ was at least once expanded, and $n$ is not marked for deletion, and $n$ is a candidate for deletion, then
       i. Mark $n$ for deletion and delete $G_{cl}(n)$.
       ii. For all successor $n'$ of $n$ in $FRONTIER$, mark the operator $n' \rightarrow n$ as 'used'.
   (b) If $n$ is marked for deletion and $G_{op}(n) = \emptyset$, then remove $n$ from $FRONTIER$.
7. Go back to step 2
```

Table 1: FS-NAMOA*, a frontier multiobjective search algorithm.

to n_1 and n_3 are generated, setting $G_{op}(n_1) = \{(1,2)\}$ and $G_{op}(n_3) = \{(2,1)\}$. Since $(0,0)$ dominates both open vectors, s is candidate for deletion and $G_{cl}(s)$ deleted. Since $G_{op}(s) = \emptyset$, s is in fact deleted from memory, and the corresponding operators in n_1 and n_3 marked as 'used'. Figure 1 (b) shows a trace of FS-NAMOA* on this example, including nodes in FRONTIER, and a graph showing the state of known paths at each iteration. Each path $(\vec{g}, n)$ is represented by its cost in cost space ($\times \equiv$ deleted; $\circ \equiv$ open; $\bullet \equiv$ closed) and labeled by its destination node n. This way the state of paths and the values of the G_{op} and G_{cl} sets can be easily told.

Assuming ties between nondominated paths are broken arbitrarily, at iteration 2 path $(n_1, (1,2))$ is selected for expansion. New paths to n_2 and n_4 are generated. However, n_1 is not a candidate for deletion, since a new nondominated path might still reach n_1 from n_3 or some of its descendants. Therefore, the closed path must remain in memory, i.e. $G_{cl}(n_1) = \{(1,2)\}$. At iteration 3 the only nondominated path to n_3 is selected, n_3 is candidate for deletion and, in fact, deleted, marking the corresponding operators as 'used' in n_4 and n_6. Notice that no new nondominated path can ever reach n_1 since the cost of the closed path to n_1 now dominates the cost of all other open paths. Node n_1 becomes candidate for deletion and is deleted. The two cost vectors of paths leading to n_4 dominate all other open paths. Therefore, n_4 is candidate for deletion. However it has never been expanded and must remain in memory. At iteration 4 a path to n_4 is selected for expansion and deleted from memory. At iteration 5 the other path to n_4 is selected and deleted, and the whole node can be deleted as well. At iteration 6 only the paths to γ_1 are nondominated. Path $(\gamma_1, (4,5))$ could be selected. Since it is a solution path, it would be recorded in $COSTS$ and $GOALN$. All open paths dominated by $(4,5)$ would be filtered, i.e. removed from $OPEN$. At iteration 7 $(\gamma_1, (5,4))$ would be the only remaining open path, selected, and recorded as solution. The algorithm would then terminate with the guarantee that node γ_1 is reached by nondominated solutions with costs in $((4,5),(5,4))$.

4 Properties

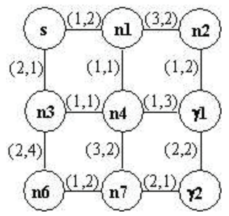
This section proves that each run of FS-NAMOA* mimics the workings of NAMOA*. Therefore, the algorithm terminates with the same sets of nondominated costs and goal nodes.

Result 1 *If costs are positive and n is marked for deletion, then all nondominated paths to n have been generated.*

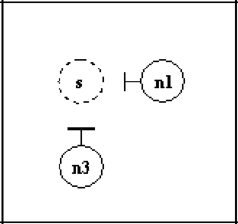
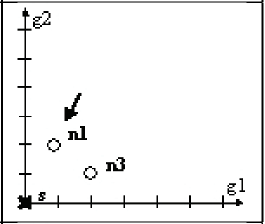
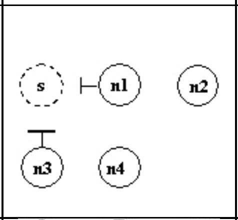
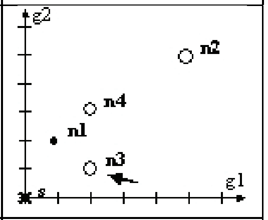
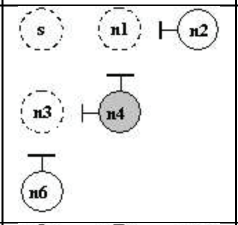
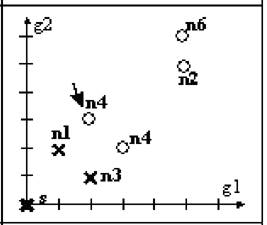
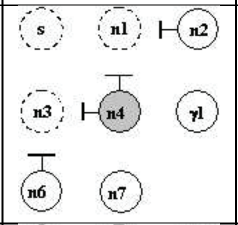
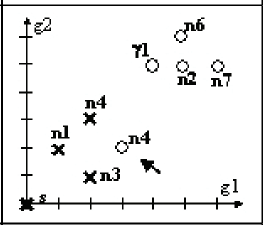
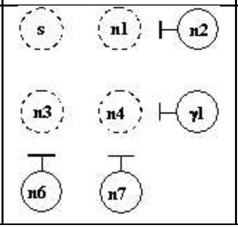
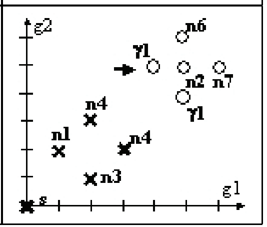
Proof. The proof is trivial from the definition of 'candidate for deletion'. Let's assume n is marked at iteration k. If a new node reaches n at future iterations, it must be some extension of an alternative $(n', \vec{g}')$ open at iteration k. Let $\vec{g}'' > \vec{0}$ be the cost of such extension. However, if n is marked, then $\exists \vec{g} \in G(n)\ \vec{g} \preceq \vec{g}' \Rightarrow \vec{g} \prec \vec{g}' + \vec{g}''$.□

Result 1 replaces the monotonicity requirement of heuristics in scalar frontier search with a monotonicity requirement for the real cost function $\vec{g}(P)$. The following results are analogous to those presented in [Korf *et al.*, 2005].

Result 2 *FS-NAMOA* never regenerates a node that has been deleted.*



(a) Sample problem for FS-NAMOA*.

It.	Frontier	Paths
2		
3		
4		
5		
6		

(b) Frontier nodes and paths in cost space $\vec{g} = (g_1, g_2)$. Symbol ⊢ denotes 'used operator', dashed nodes are deleted.

Figure 1: Sample problem and solution.

Proof. Let us assume, for the purpose of contradiction, that n is the first such node to be regenerated after a new path is found from some other node n' to n. If n was deleted, then: a) n was candidate for deletion, b) n was at least once expanded, c) $G_{op}(n) = \emptyset$. If n' was deleted before n, then n' must have been regenerated in order to be expanded again regenerating n. However, this contradicts the assumption that n was the first node to be regenerated. Then, it must be that n' was never deleted. Since n was at least once expanded, n' must have been in $FRONTIER$ (recall the graph is undirected) when n was marked for deletion, and the operator from n' to n marked as 'used'. Therefore, no path leading to n' and selected for expansion will ever regenerate n, contradicting the assumption that n is regenerated.□

Result 3 *With the same tie-breaking rule, FS-NAMOA* expands the same nodes in the same order as NAMOA*.*

Proof. We shall prove by induction the stronger result that, after each iteration, the set of open alternatives is the same for both algorithms. The proposition is true at iteration 0, when $(s, \vec{0})$ is the only open alternative. Let us assume that both algorithms selected the same paths up to iteration k, and that both have exactly the same set of open alternatives. Since both algorithms share the tie-breaking rule, they will select the same open alternative $(n, \vec{g})$ for expansion.

The following scenarios are possible for NAMOA* for each succesor m reached by cost vector $\vec{g}_m = \vec{c}(n, m) + \vec{g}$,

1. $m \in \Gamma$. All open alternatives dominated by $\vec{g}_m$ are eliminated.

2. $m \notin \Gamma$. The set F_m of estimates of new paths to m not dominated by $COSTS$ is calculated. In case: a) m is a new node. If $F_m \neq \emptyset$, then m is put in the search graph and g_m in $G_{op}(m)$; b) m is known and $\vec{g}_m \in G(m)$. No changes are made to $G_{op}(m)$ nor $G_{cl}(m)$; c) m is known and $\vec{g}_m \notin G(m)$. Dominated vectors are eliminated from $G(m)$. If $F_m \neq \emptyset$, then g_m is put in $G_{op}(m)$.

FS-NAMOA* would behave the same in situations 1 and 2c. In situation 2a the new node will be unconditionally placed in $FRONTIER$, but the new open alternative would be added under the same condition. Finally, situation 2b is ignored by FS-NAMOA*, but results in no change in the set of open alternatives. In summary, both algorithms will perform the same changes in the set of open alternatives. Since both algorithms terminate when the set of open alternatives is empty, they will perform the same sequence of path selections and expansions.□

Results 2 and 3 do not require the heuristic function to be monotone. This means FS-NAMOA* mimics NAMOA* even when non-monotone heuristics are used. Note that a single-objective version of FS-NAMOA* amounts to a version of frontier search A* that finds all optimal solutions with general heuristics. Nodes selected for expansion would be deleted only when they achieve the minimum value of $g(n)$ among open nodes. This ensures the optimal path to deleted nodes has been found.

Table 2: Summary of test results.

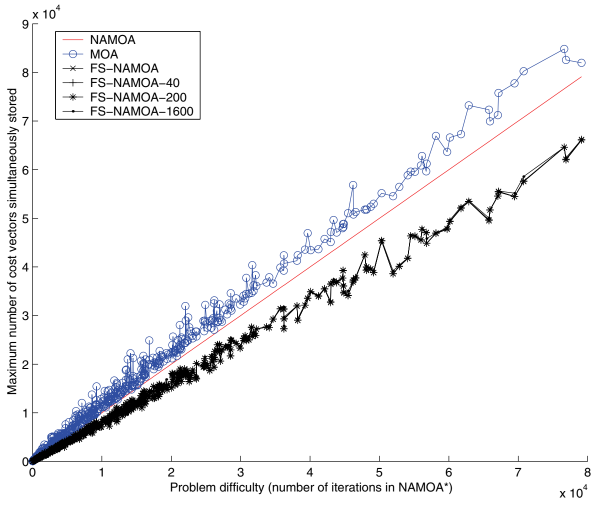
Algorithm	average	σ	min	max
FS-NAMOA*				
Cost vectors	12201.41	12610.67	38	66075
Time (s)	582.65	1085.22	0	7669
FS-NAMOA*40				
Cost vectors	12213.29	12610.91	46	66086
Time (s)	17.99	31.61	0	212
FS-NAMOA200				
Cost vectors	12259.34	12625.39	46	66212
Time (s)	6.48	10.17	0	76
FS-NAMO1600				
Cost vectors	12363.28	12670.43	46	66217
Time (s)	3.76	5.12	0	32
NAMOA*				
Cost vectors	15098.69	15514.82	47	79622
Time (s)	2.45	3.14	0	20
MOA*				
Cost vectors	17161.78	16830.14	56	84830
Time (s)	2.15	2.82	0	27

5 Experimental tests

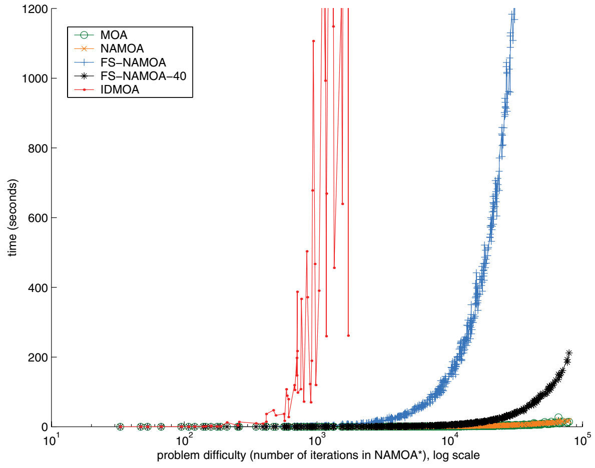
A set of 500 experimental tests were carried out in different square grids of nodes of size 101×101 with random costs in $[1, 10]$ and two objectives. Grid distance was used as heuristic for all objectives. The tests reported in this section compare the performance of FS-NAMOA* against other standard multiobjective search algorithms and are intentionally of relatively small size.

Figures 5 (a) and (b) show the performance of MOA*, NAMOA*, and FS-NAMOA* in memory and time (the latter including IDMOA*) against problem difficulty. The number of iterations (i.e. paths considered) by NAMOA* was taken as a measure of problem difficulty. Note that the x-axis in figure 5 (b) is in logarithmic scale to allow comparison to IDMOA*. Statistics are summarized in table 2.

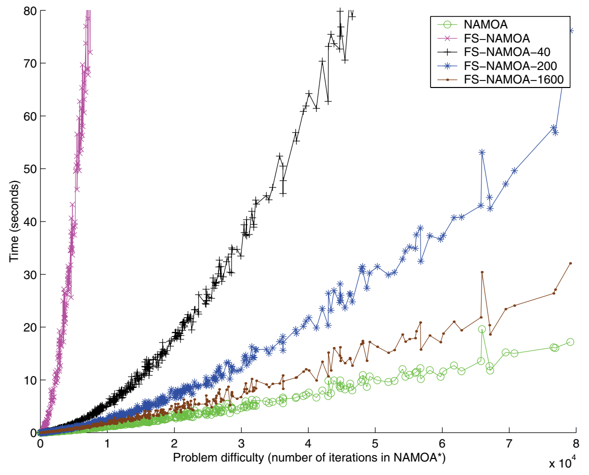
MOA* and NAMOA* require storage for 140% and 124% of the cost vectors needed by FS-NAMOA* respectively. MOA* was the fastest on average, though the speed of MOA* and NAMOA* is barely distinguishable at this scale. The extra 'update frontier' step in FS-NAMOA* places a heavy time overhead. However, the time required by the algorithm is well below the exponential time requirements of the linear space algorithm IDMOA*. In fact, this algorithm could solve only the simplest problems in less than 20 minutes. This overhead can be significantly reduced delaying frontier updates. If updates were performed evey k iterations, the overhead could be reduced by $1/k$. If the maximum branching factor of the problem is b, this will increment the memory requirements by at most kb cost vectors, which is normally a tiny fraction of the maximum memory usage. Figure 5(c) and table 2 show the time requirements for $k = 40, 200$ and 1600 with only a 1.3% increase in average memory requirements over plain FS-NAMOA (labeled FS-NAMOA-40, 200 and 1600 respectively). This suggests that only a few updates are responsible for most of the memory savings, with additional updates becoming increasingly less cost-effective.



(a) Memory requirements (cost vectors).



(b) Time requirements.



(c) Timings of FS-NAMOA with delayed updates.

Figure 2: Memory and time requirements.

6 Conclusions and future work

The paper considers the extension of frontier search to the multiobjective framework. Regrettably, the use of monotone heuristics does not allow a trivial extension of scalar frontier search to the multiobjective case. The determination of safe criteria for the deletion of expanded nodes and cost vectors turns out to be the fundamental issue in this extension. A set of such criteria have been proposed and proven to preserve the admissibility of multiobjective search even when non-monotone heuristics are used. Experimental results show that memory savings are significative enough to make them an interesting alternative. The time overhead of the deletion criteria can be greatly reduced with minimum loss in memory savings. Therefore, multiobjective frontier search fills a gap between existing best-first and depth-first strategies in their memory-time tradeoff.

Future work includes more formal development of the algorithm, and the necessary extensions to make it a more general search strategy. A comparison of all existing multiobjective search strategies over a representative set of problem domains is also an important future work.

References

[Alechina and Logan, 2001] Natasha Alechina and Brian Logan. State space search with prioritised soft constraints. *Applied Intelligence*, 14(3):263–278, 2001.

[Dasgupta *et al.*, 1999] Pallab Dasgupta, P.P. Chakrabarti, and S.C. DeSarkar. *Multiobjective Heuristic Search*. Vieweg, Braunschweig/Wiesbaden, 1999.

[Harikumar and Kumar, 1996] S. Harikumar and Shashi Kumar. Iterative deepening multiobjective A*. *Information Processing Letters*, 58:11–15, 1996.

[Hart *et al.*, 1968] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Systems Science and Cybernetics SSC-4*, 2:100–107, 1968.

[Korf *et al.*, 2005] Richard Korf, Weixiong Zhang, Ignacio Thayer, and Heath Hohwald. Frontier search. *Journal of the ACM*, 52(5):715–748, September 2005.

[Korf, 1985] Richard E. Korf. Iterative-deepening A*: an optimal admissible tree search. In *Proc. of the IX Int. Joint Conf. on Artificial Intelligence (IJCAI'85)*, pages 1034–1036, 1985.

[Korf, 1993] Richard E. Korf. Linear-space best-first search. *Artificial Intelligence*, 62:41–78, 1993.

[Mandow and Pérez de la Cruz, 2005] L. Mandow and J.L. Pérez de la Cruz. A new approach to multiobjective A* search. In *Proc. of the XIX Int. Joint Conf. on Artificial Intelligence (IJCAI'05)*, pages 218–223, 2005.

[Refanidis and Vlahavas, 2003] Ioannis Refanidis and Ioannis Vlahavas. Multiobjective heuristic state-space search. *Artificial Intelligence*, 145:1–32, 2003.

[Stewart and White, 1991] Bradley S. Stewart and Chelsea C. White. Multiobjective A*. *Journal of the ACM*, 38(4):775–814, October 1991.