A note on the complexity of some multiobjective A* search algorithms [1]

Lawrence Mandow and **José-Luis Pérez de la Cruz**[2]

Abstract. This paper studies the complexity of two different algorithms proposed as extensions of A* for multiobjective search: MOA* and NAMOA*. It is known that, for any given problem, NAMOA* requires the consideration of no more alternatives than MOA* when provided with the same heuristic information.

In this paper we show that, in fact, expansions performed by MOA* can be many more than those demanded by the problem, and hence than those performed by NAMOA*. More specifically, we show a sequence of problems whose size grows linearly such that the number of expansions performed by NAMOA* grows also linearly, but the number of expansions performed by MOA* grows exponentially. Therefore, there are problems where NAMOA* performs exponentially better than MOA*.

1 INTRODUCTION

The problem of searching for shortest paths with several conflicting objectives has important applications in areas like transportation, path planning and domain independent planning. These problems rarely have a single optimal solution. Most frequently, a set of nondominated (Pareto-optimal) solutions can be found, each one presenting a particular trade-off between the objectives under consideration. The Artificial Intelligence community has contributed several extensions of the A* algorithm to the multiobjective case. These include MOA* [5] and NAMOA* [2].

Since the number of different nondominated solutions can grow exponentially with the size of the problem, it is clear that time requirements also grow exponentially in the worst case [1]. However, in many cases the number of nondominated solutions grows polynomically and it is desirable that in these cases the search algorithm do not add more complexity that the one demanded by the problem. More detailed analyses on the complexity of multiobjective search, and on the relative performance of NAMOA* and MOA* are important open research questions.

For monotone heuristics, it has been proven [3] that NAMOA*, so to speak, does not perform more expansions that those needed by the problem. On the other hand, when running MOA*, consistency and monotonicity of the heuristic "... are not sufficient to guarantee that nodes will not need to be reopened." [5, p. 806]

In this paper we show that, in fact, expansions performed by MOA* can be many more than those demanded by the problem (and hence than those performed by NAMOA*). More specifically, for unbounded, integer arc costs we show a sequence of problems whose size grows linearly such that the number of expansions performed by NAMOA* grows also linearly, but the number of expansions performed by MOA* grows exponentially.

In the following section we recall the definitions and previous results needed for this paper. Section 3 presents a class of multiobjective search problems and the main result. Finally, our results are summarized.

2 DEFINITIONS AND PREVIOUS RESULTS

Multiobjective problems try to optimize simultaneously a number of possibly conflicting objectives. The cost of alternatives is denoted by vectors $\vec{f} \in \mathbb{R}^q$, $q > 1$, where each component of a cost vector represents the value of a different objective. Cost vectors induce a partial order preference relation $\prec$ called *dominance*, defined as follows: for all $\vec{f}$, $\vec{f'} \in \mathbb{R}^q$, $\vec{f} \prec \vec{f'}$ iff for all i $f_i \leq f'_i$ and $\vec{f} \neq \vec{f'}$, where f_i denotes the i-th element of vector $\vec{f}$.

Given a set of vectors X, we shall define $nd(X)$ the set of nondominated vectors in set X in the following way: $\mathrm{nd}(X) = \{\vec{x} \in X \mid \nexists \vec{y} \in X \quad \vec{y} \prec \vec{x}\}$.

The *lexicographic order* ($\prec_L$) is a total order relation defined over vector costs,

$\forall \vec{x}, \vec{y} \in \mathbb{R}^q \quad \vec{x} \prec_L \vec{y} \quad \Leftrightarrow \exists j \quad \forall i < j \quad (x_i = y_i \wedge x_j < y_j)$

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[3] $\vec{c}(n, n') \in \mathbb{R}^q$. We define a path in G as any sequence of nodes $P = (n_1, n_2, ...n_k)$ such that for all $i < k$, $(n_i, n_{i+1}) \in A$. The cost of each path $\vec{c}(P)$ is the sum of the cost vectors of its component arcs. The multiobjective search problem can be stated as follows: given a start node $s \in N$, and a set of goal nodes $\Gamma \subseteq N$, find the set of non-dominated cost paths in G from s to nodes in Γ.

For the sake of simplicity, in the following discussion we shall allow arcs to be labelled with sets of vectors as long as these are nondominated. Thus, an arc (n, n') labelled with nondominated costs $\vec{c}_1$ and $\vec{c}_2$ denotes that the same transition can be achieved by different actions with different (nondominated) effects in cost space.

2.1 Algorithm MOA*

The application of the ideas behind A* to multiobjective search problems resulted in the algorithm MOA* (Multi-Objective A*) [5]. This algorithm was proven to find the set of all nondominated solutions in finite graphs, whenever an optimistic (admissible) heuristic is used. If arc costs are lower bounded and positive, the property also holds

[1] This work is partially funded by/Este trabajo está parcialmente financiado por: grant TIN2009-14179 (Spanish Government, "Plan Nacional de I+D+i"), and Consejería de Innovación, Ciencia y Empresa. Junta de Andalucía (España), grant P07-TIC-03018

[2] Universidad de Malaga, Spain, email: {mandow, perez}@lcc.uma.es

[3] Published proofs of termination for algorithms MOA* and NAMOA* state this condition. However, a weaker condition is enough: that for every path P, the cost $\vec{c}(P)$ increases without bound when the length of P increases.

for infinite graphs [4]. Analogously to A*, MOA* uses two sets of nodes, *OPEN* and *CLOSED*, to control the search. The main difference lies in the generalization of scalar node functions g, h, f to functions G, H, F that return sets of vectors for each node. Additionally, $\mathrm{LABEL}(n', n)$ keeps the set of vectors in $G(n')$ that arise from a path to n' coming from n.

The set $G(n)$, computed for each known node, contains the set of nondominated cost vectors of paths found to n. In general, heuristic estimates will involve a set of vectors $H(n)$ for each node n, estimating cost vectors of paths from n to each goal node. The set $F(n)$ is the analogue in MOA* of the evaluation function $f(n)$ used by A*,

$F(n) = \mathrm{nd}\{\vec{g} + \vec{h} \mid \vec{g} \in G(n) \wedge \vec{h} \in H(n)\}$

In each step, MOA* computes the set *ND* of nondominated nodes in *OPEN*, i. e., the nodes in *OPEN* that have at least one F value that is not dominated by any F value of another node in *OPEN*. If *ND*=$\varnothing$, the algorithm terminates; otherwise, a node n is selected from *ND* and expanded.

The expansion of n entails generating every successor n' of n and assigning to $G(n')$ suitable values. If n' is a new node, then it is included in *OPEN* and $G(n')$ and $\mathrm{LABEL}(n', n)$ store all the paths expanded from n. If n' was previously generated, MOA* checks if a new nondominated value of $G(n')$ has been computed at the present step; if it is so, then the new value is stored in $G(n')$ and $\mathrm{LABEL}(n', n)$ and, if n' was in *CLOSED*, it must be moved again to *OPEN*.

In MOA* all paths reaching a node n are expanded simultaneously once n is selected. Therefore, all paths reaching a single node at a given time are either simultaneously open or closed.

For the sake of completeness, the pseudocode for MOA* is shown in Figure 1, slightly adapted from the original paper [5].

2.2 Algorithm NAMOA*

More recently, a new approach to multiobjective A* denoted NAMOA* was presented [2]. It can be proven that the new algorithm finds the set of all nondominated solutions under the same assumptions as MOA*. It is based on the idea of *path* or *cost expansion*, instead of that of *node expansion*.

We shall denote by $\vec{g}(P)$ the cost vector of each individual path stored in the search graph. A set of heuristic estimates $H(n)$ is defined as in MOA* (see previous section). Therefore, for each path P_{sn} from s to n with cost $\vec{g}(P) = \vec{g}_P$, there will be a set of heuristic evaluation vectors, $F(P_{sn})$. This function is the analogue in NAMOA* to $f(n)$ in A*,

$F(P_{sn}) = F(n, \vec{g}_P) = \mathrm{nd}\{\vec{g}_P + \vec{h} \,|\vec{h} \in H(n)\}$

The algorithm keeps an $OPEN$ list of *partial solution paths* that can be further explored. For each node n and each nondominated cost vector $\vec{g} \in G_{op}(n)$, there is a corresponding tuple $(n, \vec{g}, F(n, \vec{g}))$ in $OPEN$. Initially, $(s, \vec{g}_s, F(s, \vec{g}_s))$ is the only tuple in $OPEN$.

For each node n in SG, $G_{cl}(n)$ and $G_{op}(n)$ denote the sets of nondominated cost vectors of paths reaching n that have and have not been explored yet respectively (i.e. closed and open). Each cost vector in these sets labels one or more arcs in the graph from n to their parents.

At each iteration, the algorithm considers the expansion of an open tuple $(n, \vec{g}, F)$ that stands for a single partial solution path from s to n with cost $\vec{g}$. Thus, an important difference between NAMOA* and MOA* is that the former expands paths individually and distinguishes between open and closed paths reaching a given node (the $G_{cl}(n)$ and $G_{op}(n)$ sets). On the other hand, MOA* expands *nodes* at each iteration and all paths reaching a given node (the $G(n)$ sets) are either simultaneously open or closed.

[4] This property of the algorithm is frequently referred to as *admissibility* or *optimality*

1. INITIALIZE a set $OPEN$ with the start node s, and empty sets, $GOALN$, $COSTS$, $CLOSED$ and $LABEL$.
2. CALCULATE the set ND of nodes n in $OPEN$ such that at least one estimate $\vec{f} \in F(n)$ is not dominated by the estimates of other open nodes or by $COSTS$.
3. If ND is empty, then
   —Terminate returning the set of solution paths that reach nodes in $GOALN$ with costs in $COSTS$.
   else
   —Choose a node n from ND using a domain-specific heuristic, breaking ties in favor of goal nodes, and move n from $OPEN$ to $CLOSED$.
4. Do bookkeeping to maintain accrued costs and node selection function values.
5. IDENTIFY SOLUTIONS. If $n \in \Gamma$, then
   —Include n in $GOALN$ and its current costs into $COSTS$.
   —Remove dominated costs from $COSTS$.
   —Go back to step 2.
6. EXPAND n and examine its successors. For all successors nodes m of n do:
   (a) If m is a newly generated node, then
      i. Establish a pointer from m to n.
      ii. Set $G(m) = LABEL(m, n)$, the set of nondominated costs of paths reaching m from n discovered so far.
      iii. Compute $F(m)$.
      iv. Add m to $OPEN$.
   (b) Otherwise, m was previously generated, so do the following,
      i. If any potentially nondominated paths to m have been discovered, then, for each one, do the following.
      —Ensure that its cost is in $LABEL(m, n)$, and therefore in $G(m)$.
      —If a new cost was added to $G(m)$ then, purge from $LABEL(m, n)$ dominated costs, and if m was in $CLOSED$, then move it to $OPEN$.
7. Go back to step 2.

Figure 1. The MOA* algorithm.

In the descriptions of both algorithms above we have omitted some steps concerning the filtering of F and G values by solution costs found during the execution of the algorithm. These steps do not affect the reasoning in the following sections.

For the sake of completeness, the pseudocode for NAMOA* is shown in Figure 2, slightly adapted from the original paper [2].

2.3 Heuristics

A multiobjective heuristic function $H(n)$ is *monotone* if for all arcs (n, n') in the graph, the following condition holds: for all $\vec{h}' \in H(n')$ there exists $\vec{h} \in H(n)$ such that $\vec{h} \preceq \vec{c}(n, n') + \vec{h}'$.

A multiobjective heuristic function $H(n)$ is *consistent* if for all pairs of nodes n, n' in the graph, for all nondominated path between them $P = (n, \ldots, n')$, and for all heuristic cost vector $\vec{h}' \in H(n')$, the following condition holds: there exists $\vec{h} \in H(n)$ such that $\vec{h} \preceq \vec{c}(P) + \vec{h}'$

It can be shown that $H(n)$ is consistent if and only if it is monotone [5, Lemma 18], and that if $H(n)$ is monotone and for all goal node n ,$H(n) = \vec{0}$, then it is also admissible [5, Lemma 19].

The following theorem has also been proven in a previous paper [2]:

Theorem 1 *If the heuristic function $H(n)$ is monotone, a necessary condition for NAMOA* to select a path $P = (s, \ldots, n)$ for expansion is that P be a nondominated path to n.*

Notice that this entails that if a path $P = (s, \ldots, n)$ is expanded by NAMOA*, then will not appear again in *OPEN*. On the contrary, concerning node expansions performed by MOA*, it is acknowledged by its authors [5, p. 806] that monotonicity and consistency "[...] are not sufficient to guarantee that nodes will not need to be reopened."

The following discussion considers problems where $\forall n \quad H(n) = \{\vec{0}\}$, which is trivially monotone.

3 MAIN RESULT

For each integer $n, n > 3$, let us define a graph D_n in the following way:

- D_n has $n + 1$ nodes, labelled $0, 1, \ldots, n$.
- For each pair of integers i, j, $n \geq i > j > 0$, there exists an arc (i, j). There is also an arc from 1 to 0. Hence there are exactly $\frac{n(n-1)}{2} + 1$ arcs in D_n.
- Each arc (i, j), $n \geq i > j \geq 0$, has a vectorial cost $\vec{b}(i, j)$ given by the following recursion: (i) $\vec{b}(i, i-1) = (1, 1)$ if $i \notin \{1, n\}$; $\vec{b}(n, n-1) = (2^{n-2}, 1)$; $\vec{b}(1, 0) = (n-1+2^{n-2}, n-1+2^{n-2})$; (ii) for $1 < k < i$, $\vec{b}(i, i-k) = b(i, i-(k-1)) + (1, 2^{i-k-1}+1)$.
- Additionally, each arc (n, j), $n > j > 0$, has another vectorial cost $\vec{a}(n, j)$ given by $\vec{a}(n, j) = (j, n - 1 + 2^{n-2})$.
  Therefore, there are exactly $\frac{n(n-1)}{2} + n = \frac{n(n+1)}{2}$ arc costs.

The graph D_5 is displayed in Figure 3 (notice that a similar sequence of graphs is presented by Martelli [4].)

Now let us define a multiobjective search problem P_n on D_n in the following way:

- The start node is n.
- The goal node is 0.
- The heuristic function returns $H(n) = \{\vec{0}\}$ for every node n.
- When there are several nondominated open paths/nodes, lexicographic order is used to select one of them.

Let us now describe informally the execution of NAMOA* and MOA* when they solve the problem P_5.

At the first step, both NAMOA* and MOA* expand the start node n_5 and generate nodes n_4, n_3, n_2, n_1. Each node has two costs given by the two labels of arcs (n_5, n_i). The costs are displayed in Figure 4. Notice that in this figure costs a_i and b_i are those of the paths ending at n_i (at this step, just one path with two different costs.)

Then NAMOA* will select a nondominated path. There are exactly two, a_1 and b_4. Lexicographical order will select a_1 and the solution with cost (13, 24) will be generated. There are now two nondominated paths, a_2 and b_4. Lexicographical order will select a_2; its expansion generates a cost to n_1 that is dominated by a_1 and hence discarded. Something similar will happen with the next expansions, those of a_3 and a_4. Finally it is the turn for b_4. Its expansion generates the cost b'_3 =(9,2) for n_3; since it dominates b_3, b_3 is discarded and b'_3 is saved. Now it is the turn for b'_3, which generates b'_2 =(10,3), that replaces b_2 and is expanded generating b'_1 =(11,4), that replaces b_1 and is expanded generating a new solution with cost (23,16). The

1. CREATE:
   —An empty search graph SG, and place s as its root.
   —List of alternatives, $OPEN$ = $\{(s, \vec{g}_s, F(s, \vec{g}_s))\}$.
   —Two empty sets, $GOALN$, $COSTS$.
2. CHECK TERMINATION. If $OPEN$ is empty, then backtrack in SG from the nodes in $GOALN$ and return the set of solution paths with costs in $COSTS$.
3. PATH SELECTION. Select an alternative $(n, \vec{g}_n, F)$ from OPEN with $\vec{f} \in F$ non-dominated in $OPEN$, i.e., forall $(n', \vec{g}_{n'}, F') \in OPEN$ it does not exist $\vec{f}' \in F'$ such that $\vec{f}' \prec \vec{f}$.
   Delete $(n, \vec{g}_n, F)$ from $OPEN$, and move $\vec{g}_n$ from $G_{op}(n)$ to $G_{cl}(n)$.
4. SOLUTION RECORDING. If $n \in \Gamma$, then
   —Include n in $GOALN$ and $\vec{g}_n$ in $COSTS$.
   —Eliminate from $OPEN$ all alternatives (x, g_x, F_x) such that all vectors in F_x are dominated by $\vec{g}_n$ (FILTERING).
   —Go back to step 2.
5. PATH EXPANSION: If $n \notin \Gamma$, then for all successors nodes m of n do:
   (a) Calculate the cost of the new path found to m: $\vec{g}_m = \vec{g}_n + \vec{c}(n, m)$.
   (b) If m is a new node
      i. Calculate $F_m = F(m, \vec{g}_m)$ filtering estimates dominated by $COSTS$.
      ii. If F_m is not empty, put $(m, \vec{g}_m, F_m)$ in $OPEN$, and put $\vec{g}_m$ in $G_{op}(m)$ labelling a pointer in SG from m to n.
      iii. Go to step 2.

      else (m is not a new node),
      —If $\vec{g}_m \in G_{op}(m)$ or $\vec{g}_m \in G_{cl}(m)$: label with $\vec{g}_m$ a pointer in SG from m to n, and go to step 2.
      —If $\vec{g}_m$ is non-dominated by any cost vectors in $G_{op}(m) \cup G_{cl}(m)$ (a path to m with new cost has been found), then:
      i. Eliminate from $G_{op}(m)$ and $G_{cl}(m)$ vectors dominated by $\vec{g}_m$.
      ii. Calculate $F_m = F(m, \vec{g}_m)$ filtering estimates dominated by $COSTS$.
      iii. If F_m is not empty, put $(m, \vec{g}_m, F_m)$ in $OPEN$, and put $\vec{g}_m$ in $G_{op}(m)$ labelling a pointer in SG from m to n.
      iv. Go to step 2.
      —Otherwise: go to step 2.

Figure 2. The NAMOA* algorithm.

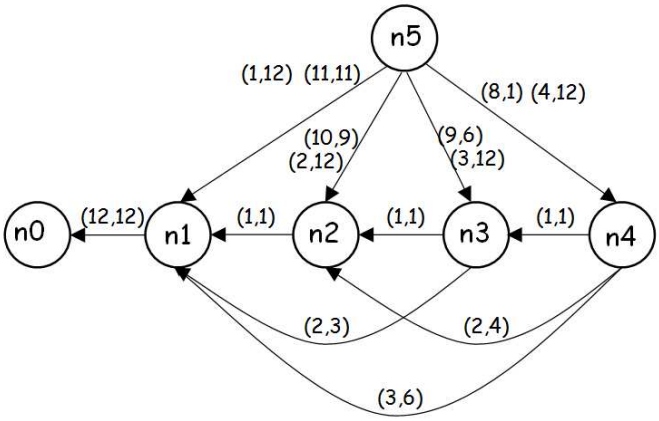

Figure 3. The graph D_5.

algorithm now terminates.

Let us now consider the execution of MOA* departing from the situation displayed in Figure 4. There are exactly two nondominated open nodes: n_1, because a_1 is a nondominated cost, and n_4, because b_4 is a nondominated cost. Lexicographical order will select n_1; its expansion generates two solutions, with costs (13, 24) and (23,23). There are now two nondominated open nodes: n_2, because a_2 is a nondominated path, and n_4, because b_4 is a nondominated path. Lexicographical order will select n_2; its expansion generates two costs to n_1: (3,13), which is dominated by a_1 and hence discarded, and b'_1 =(11,10), which dominates b_1. Therefore the new cost b'_1 =(11,10) is saved for n_1 and b_1 discarded; and, since a new cost for n_1 has been found, n_1 is again in *OPEN* with all its nondominated costs. But one of these costs, a_1, is again the best cost; so n_1 is again expanded, generating again the solution with cost (13, 24) and a new solution with cost (23,22), which dominates (23,23) and therefore replaces it. If the reader is patient enough, he can check that n_1 will be expanded eight times (once for each possible path reaching n_1 from n_5); n_2 will be expanded four times (once for each possible path reaching n_2 from n_5); and n_3 will be expanded twice.

By expressing the above analysis in an abstract way, we can prove that MOA* will perform $O(2^n)$ expansions on P_n.

Lemma 1 *Applied to the search problem P_n, NAMOA* performs at most $2n-1$ path expansions.*

Proof: By theorem 1, NAMOA* will never expand a nondominated path to i. It is easy to check that exactly two nondominated paths arrive at each node i in D_n ($0 < i < n$): one of them comes directly from n with a cost $(i, n-1+2^{n-2})$ (in the following we will call them a-costs and a-paths.) The other nondominated path is $(n, n-1, \ldots, i)$ with a cost $(2^{n-2}, 1)+(1,1)+\ldots+(1,1) = (2^{n-2}+n-1-i, n-i)$ (in the following we will them b-costs and b-paths.) The start node has only the null path $()$. So, there are at most $2(n-1)+1$ nondominated paths and hence $2n-1$ path expansions.

Lemma 2 *Applied to the search problem P_n, MOA* performs exactly 2^{n-1} node expansions.*

Proof: we will provide a proof by induction on the size n of the problem P_n and the graph D_n.

Base case: let us consider the case $n = 4$. It is easy to check that MOA* will perform exactly 8 node expansions, namely (4, 1, 2, 1, 3, 1, 2, 1).

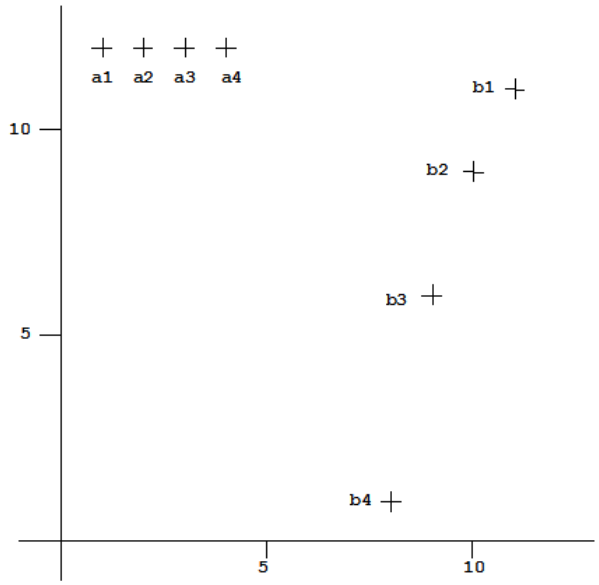

Figure 4. Path costs after expansion of $n5$.

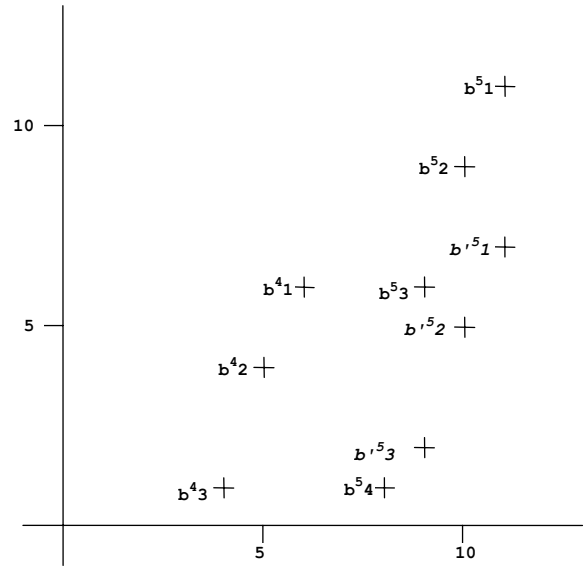

Figure 5. Path b-costs for D_4 and D_5.

Induction step: let us suppose that MOA* expands exactly 2^{n-2} nodes for P_{n-1}. We must show that MOA* expands exactly 2^{n-1} nodes for P_n. We will do so by proving that the sequence of expansions is as follows:

(i) First MOA* expands n.

(ii)Then MOA* expands the same sequence of nodes that is expanded in P_{n-1}, except for the first node $n-1$.

(iii) Then MOA* expands again the same sequence of nodes that is expanded in P_{n-1}, including $n-1$.

By proving this, the induction step is done, since the number of expansions would be $1 + 2^{n-2} - 1 + 2^{n-2} = 2^{n-1}$.

Now we must prove the assertions (i), (ii) and (iii).

Figure 5 can help to understand the argumentation that will be sketched in the following paragraphs. In this figure, b_1^4, b_2^4 and b_3^4 are the initial b-costs for nodes 1, 2 and 3 in the problem P_4 (i. e., the b-costs after the expansion of node n); b_1^5, b_2^5, b_3^5 and b_4^5 are the initial b-costs for nodes 1, 2, 3 and 4 in the problem P_4; and $b_1'^5$, $b_2'^5$ and $b_3'^5$ are the b-costs for nodes 1, 2 and 3 after the expansion of node 4.

Consider the graph D'_{n-1} obtained from D_n by removing node $n-1$ and all arcs going to or from $n-1$ and labelling node n as $n-1$. Notice that D'_{n-1} is identical to D_{n-1}, except that arcs costs are different:

$\vec{a}'_{n-1}(n-1,j) = (j, n-1+2^{n-2})$ in D'_{n-1}, vs. $\vec{a}_{n-1}(n,j) = (j, n-2+2^{n-3})$ in D_{n-1};

$\vec{b}'_{n-1}(1,0) \neq \vec{b}_{n-1}(1,0)$;

And, more importantly, $\vec{b}'_{n-1}(n-1,n-2) = \vec{b}_n(n,n-2) = (2^{n-2},1) + (1,2^{n-k-1}+1) = (2^{n-2}+1, 2^{n-3}+2)$ in D'_n, vs. $\vec{b}_{n-1}(n-1,n-2) = (2^{n-3},1)$, therefore, by the recursive definition of b-costs, for $0 < i < n-1$, $\vec{b}'_{n-1}(n-1,i) - \vec{b}_{n-1}(n-1,i) = (2^{n-3}+1, 2^{n-3}+1)$.

So, all initial b-costs of nodes $1,\ldots,n-2$ in D_n are those of D_{n-1} translated by $(2^{n-3}+1, 2^{n-3}+1)$. The situation is depicted in Figure 5 for D_5 and D_4. Notice that costs b_i^5 are the costs b_i^4 translated by $(5,5)$.

Now let us consider D_n. After the first step (expansion of node n) *OPEN* = $\{1,\ldots,n-1\}$; and (by virtue of a-costs, see Figure 4) the node selected from *OPEN* will be always the least $i \in$ *OPEN*. So, node $n-1$ will be expanded only if nodes $1,\ldots,n-2$ are not in *OPEN*. But, since initial b-costs of $1,\ldots,n-2$ are a translation of those of D_{n-1}, and the rest of costs between $1,\ldots,n-2$ have not changed, the order of expansion will be the same; therefore assertions i) and ii) are proven.

To prove assertion iii), let us consider the situation of b-costs when node $n-1$ is expanded in D_n. Then a new path $(n,n-1,i)$ is found for every i, $0 < i < n-1$, with a b-cost $b''_n(i) = \vec{b}_n(n,n-1) + \vec{b}_n(n-1,i)$. In Figure 5 the values of $b_1'^5$, $b_2'^5$ and $b_3'^5$ are the b-costs for nodes 1, 2 and 3 in D_5 after the expansion of node 4.

We will prove that (a) $b''_n(i)$ dominates the b-cost stored for i at that moment; and (b) $b''_n(i) = \vec{b}_{n-1}(n-1,i) + (2^{n-3}+1, 1)$, where $\vec{b}_{n-1}(n-1,i)$ is the initial b-cost for node i in D_{n-1}. But elementary operations yield that $b''_n(i) = (2^{n-2}+n-1-i, 2^{n-3}-2^{i-1}+n-i)$ and it is easy to check that $b''_n(i) - \vec{b}_{n-1}(n-1,i) = (2^{n-3}+1, 1)$. It remains to show that this value dominates the value stored for i before the expansion of $n-1$. But the path not traversing $n-1$ that yields the best b-value is $(n, n-2, n-3, \ldots, i)$ and its b-value is $\vec{b}_n(n,n-2) + \vec{b}_n(n-2,n-3) + \ldots + \vec{b}_n(i+1,i)$ i. e., $(2^{n-2} + n - 1 - i, 2^{n-3} + n - i)$ which is dominated by $b''_n(i)$. Now, the same reasoning made to prove assertion (ii) can be used to prove that after the expansion of node $n-1$, the same expansions made when processing D_{n-1} will be made. This finishes the (sketch) of the proof of Lemma 2.

Lemmas 1 and 2 together amount to the following:

Theorem 2 *For all $n > 3$, there exists a search problem P_n of size n such that NAMOA* performs $\Omega(n)$ path expansions and MOA* performs $\Omega(2^n)$ node expansions.*

4 CONCLUSIONS

We have compared the efficiency of two multiobjective search algorithms (MOA* and NAMOA*) when applied to certain problems. Assuming that MOA* uses lexicographic order to select one non-dominated open path/node when there are several ones, we have proven that there are cases where NAMOA* performs a number of expansions that grows linearly with the size of the problem; on the other hand, MOA* performs a number of expansions that grows exponentially.

Notice that this analysis is purely theoretical and measures performance by counting expansion operations. Space and time consumption also depend on other features of the algorithms.

Our result has been proven for lexicographical tie-breaking in MOA*. It would be interesting to find a family of cases exhibiting the same behavior but not depending on the tie-breaking rule.

ACKNOWLEDGEMENTS

We would like to thank the referees for their comments, which helped improve this paper considerably.

REFERENCES

[1] P. Hansen, 'Bicriterion path problems', in *Lecture Notes in Economics and Mathematical Systems 177*, pp. 109–127. Springer, (1979).

[2] 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)*, pp. 218–223, (2005).

[3] L. Mandow and J. L. Pérez de la Cruz, 'Comparison of heuristics in multiobjective A* search.', in *Lecture Notes in Artificial Intelligence 4177*, pp. 180–189. Springer, (2006).

[4] Alberto Martelli, 'On the complexity of admissible search algorithms', *Artificial Intelligence*, **8**, 1–13, (1977).

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