Dimensionality reduction in multiobjective shortest path search

Francisco-Javier Pulido*, Lawrence Mandow, José-Luis Pérez-de-la-Cruz

Dpto. Lenguajes y Ciencias de la Computación, Universidad de Málaga, Campus de Excelencia Internacional Andalucía Tech, Boulevar Louis Pasteur, Málaga 35. 29071, Spain

Abstract

One-to-one multiobjective search in graphs deals with the problem of finding all Pareto-optimal solution paths between given start and goal nodes according to a number of distinct noncommensurate objectives. The problem is inherently more complex than single objective graph search. Time requirements are dominated by the facts that (a) many different non-dominated labels may need to be explored for each node; (b) each new label under consideration must be checked for dominance against various subsets of previously generated labels. This paper describes how a dimensionality reduction technique can be applied to exact label-setting algorithms, reducing the number of dominance checks and allowing for much faster multiobjective search. The technique is applied to NAMOA*, a state of the art exact label-setting multiobjective search algorithm, achieving reductions in time requirements of more than an order of magnitude over problems in random grids and realistic road maps. Tests include problems with three, four, and five objectives.

© 2015 Elsevier Ltd. All rights reserved.

1. Introduction

Multiobjective (MO) shortest path problems arise in many fields, such as vehicle path planning [17,39,40], urban transportation networks [6,9], robot surveillance [7], satellite scheduling [11], routing in telecommunication networks [5], and route planning in different contexts [1,3,8,16].

The multiobjective shortest path problem involves finding the set of all Pareto-optimal solution paths in a graph from a given start node to a designated goal node (one-to-one problem), or to all other nodes in the graph (one-to-all problem), according to a number of distinct noncommensurate objectives. Several approaches have been proposed to solve this problem. These include enumerative approaches (label-setting and label-correcting), ranking algorithms, and two-phase algorithms (e.g. see [4,2,34,35,12,37,31,51]). For the one-to-one problem, where single start and goal nodes are designated in the graph, label-setting algorithms are generally the best option for arbitrary start–goal pairs in large graphs, e.g. navigation queries in road maps. Multiobjective label-setting algorithms are generally extensions of single objective ones, and particularly of Dijkstra’s algorithm and A* [14]. The latter is a variant of the former for one-to-one problems that uses distance estimates to the goal to improve search efficiency. If these estimates are lower bounds, then A* is an exact algorithm, i.e. it always returns an optimal solution. The one-to-one version of Dijkstra’s algorithm is a particular case of A* with lower bounds equal to zero.1

Hansen [13] first extended Dijkstra’s algorithm to the biobjective problem, and showed that even with two objectives the number of Pareto-optimal solution paths can grow exponentially with solution depth in the worst case. However, there are interesting classes of MO search problems where this worst-case behavior does not appear [25,28]. Martins [27] provided an exact label-setting algorithm for the general case, i.e. each label scanned by the algorithm is Pareto-optimal. Martins proposed selecting the lexicographically smallest label from the set of pending labels at each step, in order to guarantee that a Pareto-optimal label is always selected. Other authors have proposed selecting the minimum according to a linear aggregate function, which has been reported to be more efficient in practice [15,20]. Several extensions of A* to the multiobjective case have also been proposed. These include MOA* [36], TC [38], and NAMOA* [26]. The latter has been proven to be an exact algorithm when provided with lower bound estimates, and to explore an optimal number of labels. Additionally, it has been formally shown that MOA* does not always explore

1 The distance estimate functions (used by A*) are called heuristic functions in the Artificial Intelligence literature (e.g. see [10,29]). However, in this paper we adopt the Operations Research terminology, where the term “heuristic” is used to designate approximate algorithms. Notice that when distance estimates are lower bounds, the algorithms considered in this paper (A* and NAMOA*) are in fact exact algorithms.
an optimal number of labels [30], Tung and Chew also proposed using the ideal point of each node \( n \) as a lower bound estimate of the cost of all paths from \( n \) to the goal [38]. These ideal points can be efficiently calculated performing \( q \) one-to-all single objective searches from the goal node in a graph identical to the original one, but with all arcs reversed (where \( q \) is the number of objectives, and each search optimizes a different objective). Using these lower bounds, NAMOA* was shown to outperform the other MO extensions of \( A^* \) over problems in random grids and realistic road maps [21,18] with two objectives.

However, even in this case, time requirements are still an important limiting factor in the size of problems that exact algorithms can solve in practice [19,21,23]. This paper describes a dimensionality reduction technique that speeds up the time performance of multiobjective label-setting search. Dimensionality reduction was first proposed as a space-saving technique in the development of vector frontier search [24,25], a multiobjective exact label-setting algorithm that achieved important reductions in space requirements at the expense of increasing time requirements. This paper extends this technique, showing that it can be applied, under reasonable assumptions, to best-first label-setting algorithms (and to NAMOA* in particular). Results show improvements in time performance. Furthermore, the use of this technique preserves all the properties of NAMOA*. More precisely, it remains an exact algorithm when provided with lower bound estimates. In other words, the proposed technique reduces the time requirements of NAMOA*, extending the size of MO problems that can be solved exactly in practice.

This paper is organized as follows. Section 2 reviews important concepts in MO search. Section 3 describes the dimensionality reduction technique, and proves its correctness for MO search with lower bounds under reasonable assumptions. An experimental study is described in Section 4, and its results are discussed in Section 5. Finally, some conclusions and future work are outlined.

2. Multiobjective search

The multiobjective search problem can be stated as follows: let \( G \) be a locally finite directed graph \( G=\{(N,A,\vec{c}), \) of \( N \) nodes, and \( A=\{(i,j),\ldots,(i_{m},j_{m})\} \subseteq N \times N \) arcs, where \( q \) positive costs \( \vec{c}_{ij}=(c_{i1},\ldots,c_{iq}) \in \mathbb{R}^{q} \) are associated with each arc \((i,j) \in A\). Sometimes we will denote \( \vec{c}_{ij} \) as \( \vec{c}(i,j) \).

Let a path from node \( n_1 \) to node \( n_k \) be a sequence of nodes \((n_1,n_2,\ldots,n_k)\) such that \( \forall i < k \ (n_i,n_{i+1}) \in A \). Let the cost vector of a path be defined as the sum of the cost vectors of its component arcs.

Definition 1 (Adapted from [31]). The multiobjective shortest path problem consist in finding the set of all non-dominated cost paths in \( G \), with source node \( s \in N \) and target node \( \gamma \in N \). It can be formulated mathematically as the following network flow problem,

\[
\begin{align*}
\min \quad & \vec{z}(\vec{x}) = \sum_{(i,j) \in A} \vec{c}_{ij} x_{ij} \\
\text{s.t.} \quad & \sum_{(i,j) \in A} x_{ij} - \sum_{(j,i) \in A} x_{ij} = \begin{cases} 
1 & \text{if } i = s, \\
0 & \text{if } i \neq s, \gamma, \\
-1 & \text{if } i = \gamma, 
\end{cases} \\
& x_{ij} \in [0,1] \quad \text{for all } (i,j) \in A.
\end{align*}
\]

where \( \vec{x} \) is a vector of flows on the arcs, and the constraints (2) represent flow balance at the different nodes.

Definition 2. In multiobjective problems, cost vectors induce a partial order preference relation \( \prec \) called dominance,

\[
\forall \vec{y},\vec{y}' \in \mathbb{R}^{q} \quad \vec{y} \prec \vec{y}' \iff \forall i \quad y_i \leq y'_i \wedge \vec{y} \neq \vec{y}'
\]

Analogously, we define the preference relation \( \preceq \), called dominance or equality,

\[
\forall \vec{y},\vec{y}' \in \mathbb{R}^{q} \quad \vec{y} \preceq \vec{y}' \iff \forall i \quad y_i \leq y'_i \wedge \vec{y} = \vec{y}'
\]

where \( y_i \) denotes the \( i \)-th element of vector \( \vec{y} \). For any two vectors, it is not always possible to rank one as better than the other according to dominance, e.g. none of \((20,30), (30,20)\) dominates each other, but both are dominated by \((10,10)\).

Definition 3. Given a set of vectors \( X \), we define \( nd(X) \), the set of non-dominated vectors in \( X \) as

\[
nd(X) = \{ \vec{x} \in X | \forall \vec{y} \in X, \vec{y} \preceq \vec{x} \}
\]

Definition 4. We define the lexicographic order \( \prec_L \) as follows:

\[
\forall \vec{y},\vec{y}' \in \mathbb{R}^{q} \quad \vec{y} \prec_L \vec{y}' \iff \exists j \left[ jy_j < y'_j \wedge \forall i < jy_i = y'_i \right]
\]

and the preference relation \( \preceq_L \).

\[
\forall \vec{y},\vec{y}' \in \mathbb{R}^{q} \quad \vec{y} \preceq_L \vec{y}' \iff \vec{y} \prec_L \vec{y}' \vee \vec{y} = \vec{y}'
\]

Definition 5. We also define the linear aggregate order \( \prec_{ln} \) as follows:

\[
\forall \vec{y},\vec{y}' \in \mathbb{R}^{q} \quad \vec{y} \prec_{ln} \vec{y}' \iff \exists i \left[ y_i < y'_i \right] \quad 1 \leq i \leq q
\]

where \( y_i \) denotes the \( i \)-th component of vector \( \vec{y} \).

The lexicographic and linear aggregate orders are strict total orders. The lexicographic and linear aggregate optima of a set of vectors are trivially non-dominated vectors.

A basic pseudocode for algorithm NAMOA* is shown in Table 1. More details about this algorithm can be found elsewhere [26]. NAMOA* is a best-first algorithm that builds a search graph \( S_G \), rooted at the start node \( s \), to store all non-dominated paths found to each node. Each node in the search graph has two sets of labels: \( G_{opt}(n) \) denotes the set of cost vectors of paths reaching \( n \) that can be further explored; \( G_c(n) \) denotes the set of cost vectors that have already been expanded, and we refer to them as closed or permanent. The algorithm uses a distance estimate function \( h(n) \) that returns a set of the cost estimates of the cost of all non-dominated paths from node \( n \) to the goal. When the cost estimates lower bound real costs, NAMOA* is an exact algorithm. In this work we are concerned with the case where each node has a single lower bound, which we denote as \( h(n) \). Let \( \vec{g}(P_m) \) be the cost of some path \( P_m \) reaching some node \( n \) from the start node. Then, \( \vec{f}(P_m) = \vec{g}(P_m) + h(n) \) is an estimate of the cost of an extension of such path to the goal node.

For each unexplored label \((n,g),g \in N, \vec{g}, \vec{g} \in G_{opt}(n)\), an extended label \((n, \vec{g}, \vec{f}) \) is calculated such that \( \vec{f} = \vec{g} + h(n) \). All such extended labels are kept in an OPEN set. At each iteration, an extended label with a non-dominated \( \vec{f} \) in OPEN is selected for expansion. By abuse of language, we will sometimes refer to extended labels simply as labels. If \( n \) is the goal node, then \( \vec{f} \) is stored in COSTS, the set of non-dominated solutions costs. Otherwise, the non-dominated label \((n, \vec{g}, \vec{f}) \) selected from OPEN is made permanent, i.e., \( g \) is moved from \( G_{opt}(n) \) to \( G_c(n) \). For each arc \((n,n')\) in the graph with cost \( \vec{c}(n,n') \), a new label is generated for
returns an optimistic estimate (lower bound) of the cost of \( NAMOA \) lexicographically, allowing for efficient algorithms (see Theorem 6.4 [26]). The following definition will be useful later on.

**Definition 6.** A single-vector multiobjective distance estimate function \( \overline{h}(n) \) is monotone if, for all arcs \((n, n')\) in the graph, the following condition holds:

\[
\overline{h}(n) \leq \overline{c}(n, n') + \overline{h}(n').
\]

### Table 1

**Algorithm NAMOA° (adapted from [26]).**

1. **CREATE:**
   - An empty search graph \( SG \), and place the start node \( s \) as its root.
   - Two sets \( G_{cl}(s) = \emptyset \) and \( G_{op}(s) = \{s\} \).
   - A list of alternatives, \( OPEN = (\{s\}, \emptyset, h(s)) \).
   - An empty set, \( COSTS \).

2. **PATH SELECTION,** if \( OPEN \) is not empty, then:
   - Select a label \((n, \overrightarrow{g}, f)\) from \( OPEN \) with \( f \) non-dominated in \( OPEN \), i.e. \( \gamma(n', \overrightarrow{g}, f) \in OPEN \) such that \( \overrightarrow{f} < \overrightarrow{f}' \).
   - Delete the selected label from \( OPEN \), and move \( \overrightarrow{g} \) from \( G_{op}(n) \) to \( G_{cl}(n) \).
   - If \( \overrightarrow{g} \in COSTS \) such that \( \overrightarrow{c} < \overrightarrow{f} \), then repeat step 2 (lazy filtering).

3. **CHECK TERMINATION.** If \( OPEN \) is empty, then backtrack in \( SG \) from the goal node \( g \) and return the set of solution paths with costs in \( COSTS \).

4. **SOLUTION RECORDING.** If \( n \) is the goal node, then:
   - Include \( \overrightarrow{g} \) in \( COSTS \).
   - Go back to step 2.

5. **PATH EXPANSION:** If \( n \) is not the goal node, then for all successor nodes \( m \) of \( n \) do:
   - Calculate the cost of the new path found to \( m \) and its evaluation, \( \overrightarrow{g}_m = \overrightarrow{g} + \overrightarrow{c}(n, m), \overrightarrow{f}_m = \overrightarrow{f} + \overrightarrow{h}(m) \).
   - If \( \overrightarrow{c} \in COSTS \), then \( \overrightarrow{f} < \overrightarrow{f}' \), then:
     - If \( m \neq SG \):
       - Set \( G_{op}(m) = (\{\overrightarrow{g}_m\}) \) and add \( (m, \overrightarrow{g}_m, \overrightarrow{f}_m) \) to \( OPEN \).
       - Label with \( \overrightarrow{g}_m \) a pointer from \( n \) to \( m \).
     - Else if \( \overrightarrow{g} \) equals some cost vector in \( G_{op}(m) \cup G_{cl}(m) \) then
       - Label with \( \overrightarrow{g}_m \) a pointer from \( n \) to \( m \).
     - Else if \( \overrightarrow{f}_m \) is not dominated by \( G_{cl}(m) \cup G_{op}(m) \), then:
       - Eliminate vectors \( \overrightarrow{g}_m \in G_{op}(m) \) such that \( \overrightarrow{g}_m < \overrightarrow{g}_m \) and their corresponding labels \( (m, \overrightarrow{g}_m, \overrightarrow{f}_m) \) from \( OPEN \).
       - Add \( (m, \overrightarrow{g}_m, \overrightarrow{f}_m) \) to \( OPEN \), \( \overrightarrow{g}_m \) to \( G_{op}(m) \), and label with \( \overrightarrow{g}_m \) a pointer from \( n \) to \( m \).
   - Go back to step 2.

These operations are computationally costly. For the case with two objectives the label sets \( G_{op}(n), G_{cl}(n) \) and \( COSTS \) can be ordered lexicographically, allowing for efficient dominance checks [33]. However, for three or more objectives there is no efficient way to check the dominance of a vector against a set.

The algorithm terminates when \( OPEN \) becomes empty. NAMOA° shares important properties with A°. If the estimate function \( \overline{h}(n) \) returns an optimistic estimate (lower bound) of the cost of any path from \( n \) to the goal, then it is guaranteed to terminate with the set of all non-dominated solution paths to the problem, i.e. NAMOA° is an exact algorithm (see Theorem 4.9 in [26]).

If the lower bound estimate satisfies the so-called monotone property, then NAMOA° explores only non-dominated labels (i.e. it is a label-setting algorithm) and is optimal in the class of admissible algorithms (see Theorem 6.4 [26]). The following definition will be useful later on.

**Definition 7.** Given a vector \( \overrightarrow{v} = (v_1, v_2, \ldots, v_n) \), its truncated vector \( \overrightarrow{t}(v) \) is vector \( \overrightarrow{v} \) without its first component, i.e. \( \overrightarrow{t}(v) = (v_2, \ldots, v_n) \).

**Definition 8.** Given a set of vectors \( X \), its associated set of truncated vectors is \( T(X) = \{ \overrightarrow{t}(x) | x \in X \} \). Notice that \( T(X) \) only contains non-dominated vectors.

In this section we propose a modification of the Op-pruning and Cl-pruning dominance checks (hopefully more efficient), and prove that, under reasonable assumptions, the new versions of the checks are equivalent to the original ones.

First, we reproduce two definitions from [25]:

**Definition 7.** Given a vector \( \overrightarrow{v} = (v_1, v_2, \ldots, v_n) \), its truncated vector \( \overrightarrow{t}(v) \) is vector \( \overrightarrow{v} \) without its first component, i.e. \( \overrightarrow{t}(v) = (v_2, \ldots, v_n) \).

**Definition 8.** Given a set of vectors \( X \), its associated set of truncated vectors is \( T(X) = \{ \overrightarrow{t}(x) | x \in X \} \). Notice that \( T(X) \) only contains non-dominated vectors.

Now we can describe a new procedure for dominance checking as follows:

**Definition 9.** Let \( X \) be a set of vectors. A vector \( \overrightarrow{v} \) is \( t \)-discarded by \( X \) when for all \( \overrightarrow{v}' \in X, v_1 \leq v_1 \), and there is \( \overrightarrow{v} \in X \) such that \( t(\overrightarrow{v}') \in T(X) \), and one of the following conditions holds:

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Algorithm NAMOA° (adapted from [26]).</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. <strong>CREATE:</strong></td>
<td>- An empty search graph ( SG ), and place the start node ( s ) as its root.</td>
</tr>
<tr>
<td>2. <strong>PATH SELECTION,</strong> if ( OPEN ) is not empty, then:</td>
<td>- Two sets ( G_{cl}(s) = \emptyset ) and ( G_{op}(s) = {s} ).</td>
</tr>
<tr>
<td>3. <strong>CHECK TERMINATION.</strong> If ( OPEN ) is empty, then:</td>
<td>- A list of alternatives, ( OPEN = ({s}, \emptyset, h(s)) ).</td>
</tr>
<tr>
<td>4. <strong>SOLUTION RECORDING.</strong> If ( n ) is the goal node, then:</td>
<td>- An empty set, ( COSTS ).</td>
</tr>
<tr>
<td>5. <strong>PATH EXPANSION:</strong> If ( n ) is not the goal node, then for all successor nodes ( m ) of ( n ) do:</td>
<td>- Include ( \overrightarrow{g} ) in ( COSTS ).</td>
</tr>
<tr>
<td>6. Go back to step 2.</td>
<td>- Go back to step 2.</td>
</tr>
</tbody>
</table>
We will prove (Theorem 1) that under Assumptions 1 and 2 the correctness of the algorithm in Table 1 is not affected when filtering and cl-pruning use t-discarding checks instead of standard dominance checks. First, we will prove two auxiliary results (Lemmas 1 and 2).

Let \( \vec{v} \) be a vector and \( X \) a set of vectors. We will write \( X \leq \vec{v} \) when \( \vec{v} \) is an upper lexicographic bound of \( X \), that is, when for every \( \vec{v}' \in X \), \( \vec{v}' \leq \vec{v} \).

**Lemma 1.** Given Assumptions 1 and 2, when a label \( l = (n, \vec{g}, \vec{f}) \) is selected from OPEN, then (i) \( G_0(n) \leq \vec{g} \); and (ii) \( \text{COSTS} \leq \vec{f} \).

**Proof.** Let us define OPEN, as the set of open labels at iteration \( t \), and assume that \( (m, \vec{g}, \vec{f}) \) was selected at iteration \( t \) of the algorithm. Then, at that moment, for all \( (n', \vec{g}', \vec{f}') \in \text{OPEN} \), we have (by Assumption 2) \( \vec{f}' \leq \vec{f} \). Let us consider now the label \( (n, \vec{g}, \vec{f}) \), selected from \( \text{OPEN}_{t+1} \) at the next iteration \( t+1 \). If \( (n, \vec{g}, \vec{f}) \in \text{OPEN} \), then obviously \( \vec{f} \leq \vec{f}' \). Otherwise, \( (n, \vec{g}, \vec{f}) \) is a new label, and then \( n \) is a successor of \( m \), and \( \vec{g} = \vec{g} + \vec{c}(m,n) \).

By Assumption 1, \( \vec{h} = \vec{h} \) if \( (m, \vec{g}, \vec{f}) \in \text{OPEN} \), and \( \vec{f} = \vec{g} + \vec{h} = \vec{g} + \vec{h}(m) \leq \vec{h} + \vec{c}(m,n) + \vec{h}(n) = \vec{g} + \vec{h}(n) = \vec{f} \) and hence \( \vec{f} \leq \vec{f}' \).

We have proved that if \( (m, \vec{g}, \vec{f}) \) is selected for expansion at iteration \( t \), and \( (n, \vec{g}, \vec{f}) \) at iteration \( t+1 \), then \( \vec{f} \leq \vec{f}' \). It trivially follows by induction that, if \( (m, \vec{g}, \vec{f}) \) is selected for expansion before \( (n, \vec{g}, \vec{f}) \), then \( \vec{f} \leq \vec{f}' \).

Now we can prove part (i) of the lemma. Let us consider the set \( G_0(n) \). If \( \vec{g} \leq G_0(n) \), then there is a closed label \( (n, \vec{g}, \vec{f}) \) at \( n \) that was selected before \( (n, \vec{g}, \vec{f}) \), so \( \vec{f} \leq \vec{f}' \). Since \( \vec{f} \leq \vec{g} + \vec{h}(n) \) and there is a successor of \( m \), then \( \vec{g} \leq \vec{g} \) and \( G_0(n) \leq \vec{g} \) as stated in part (i).

Let us consider now the set \( \text{COSTS} \). If \( \vec{f} \in \text{COSTS} \), it belongs to a label \( (\gamma, \vec{f}, \vec{f}) \) (where \( \gamma \) is the goal node) that has been already selected, so \( \vec{f} \leq \vec{f}' \), as stated in part (ii).

Notice that both Assumptions 1 and 2 were used in the proof. For example, assume that the estimate function does not satisfy the monotone property. Then, it may happen that for a label \( (m, \vec{g}, \vec{f}) \) selected for expansion before \( (n, \vec{g}, \vec{f}) \), \( h(m) \leq \vec{c}(m,n) + \vec{h}(n) \), and therefore \( \vec{f} = \vec{g} + \vec{h}(m) \leq \vec{g} + \vec{c}(m,n) + \vec{h}(n) = \vec{g} + \vec{h}(n) = \vec{f} \). Hence, it does not necessarily hold that \( \vec{f} \leq \vec{f}' \). Regarding the second assumption, suppose a linear aggregate selection order for non-dominated labels in OPEN. Then, it is clear that even if for all \( (m, \vec{g}, \vec{f}) \in \text{OPEN} \), \( \vec{f} \leq \vec{f}' \), it does not necessarily hold that \( \vec{f} \leq \vec{f}' \).

**Lemma 2.** Let \( \vec{v} \) be a vector, and \( X \) a set of vectors such that \( X \leq \vec{v} \). Then, \( \vec{v} \) is dominated by \( X \) if and only if \( \vec{v} \) is t-discarded by \( X \).

**Proof.** The "if" direction is trivial: if \( \vec{v} \) is t-discarded by \( X \), both condition (a) and condition (b) in Definition 9 imply that there exists \( \vec{v}' \in X \) such that \( \vec{v} < \vec{v}' \), hence \( \vec{v} \) is dominated by \( X \).

The "only if" direction is also immediate. Suppose \( \vec{v} \) is dominated by \( X \). Then, there exists at least a vector \( \vec{v} \in X \) such that \( \vec{v} < \vec{v} \). Suppose first that \( t(\vec{v}') \in T(X) \) and \( v_i < v_1 \). Then, it must be \( t(v_i) < t(v_1) \). Therefore \( \vec{v} \) satisfies the conditions imposed in Definition 9 (option a) to t-discard \( \vec{v} \). On the other hand, if \( v_i = v_1 \) it must be \( t(\vec{v}') < t(\vec{v}) \), and \( \vec{v} \) also satisfies the conditions imposed in Definition 9 (option b). Let us suppose now \( t(\vec{v}') \notin T(X) \). Then \( t(\vec{v}') \) is dominated by a certain \( t(\vec{v}') \), that is, \( t(\vec{v}) < t(\vec{v}') \leq t(\vec{v}) \). And, since \( \vec{v} \in X \subseteq \vec{v} \), it must be \( v_i \leq v_1 \). But if \( v_i < v_1 \), \( \vec{v} \) t-discards \( \vec{v} \) (option a); and if \( v_i = v_1 \), then \( \vec{v} \) t-discards \( \vec{v} \) (option b). So, in any case, if \( \vec{v} \) is dominated by \( X \), then \( \vec{v} \) is t-discarded by \( X \). □
Theorem 1. Under Assumptions 1 and 2, the workings of NAMOA* are unaffected if filtering and/or cl-pruning are defined as follows:

- Filtering, Discard \((n, \vec{g}, \vec{f})\) if \(\vec{f}\) is t-discarded by COSTS.
- Cl-pruning, Discard \((n, \vec{g}, \vec{f})\) if \(\vec{g}\) is t-discarded by \(G_d(n)\).

Proof. First note that by Lemma 1 when a label \(l = (n, \vec{g}, \vec{f})\) is selected from OPEN, then \(G_d(n) \leq \vec{g}\) and COSTS \(\leq \vec{f}\). Lemma 2 guarantees that in case that dominance and t-discard by a set \(X\) are equivalent. So, a new label will be cl-pruned and/or filtered exactly in the same cases.

The real advantage of t-discarding is time performance. A recent work [25] shows that when arc costs are integer in the interval \([\xi, \zeta]\), \(\xi, \zeta > 0\), and \(r = \zeta - \xi + 1\), then the worst-case number of Pareto optimal costs reaching a node at depth \(d\) with \(q\) objectives is \(O(dr^{q-1})\). This is therefore a worst-case bound on the size of the COSTS and \(G_d(n)\) sets, and in the number of dominance checks against these sets.

The use of t-discarding implies checking only against the \(T(COSTS)\) and \(T(G_d(n))\) sets. The size of vectors in these sets is \(q = q - 1\), hence the worst-case size of the \(T(COSTS)\) and \(T(G_d(n))\) sets is only \(O(dr^{q-2})\).

### Table 2

Size of relevant sets of labels for random grid problems solved by NAMOA* (averaged over 5 random problems for each solution depth \(d\)). Magnitudes \(\sum_{n} (G_d(n))\) and \(\sum_{n} (T(COSTS))\) are only the same for NAMOA* and NAMOA*.

<table>
<thead>
<tr>
<th>(q)</th>
<th>(d)</th>
<th>Max OPEN</th>
<th>(\sum_{n} (G_d(n)))</th>
<th>(\sum_{n} (T(COSTS)))</th>
<th>%</th>
<th>(\sum_{n} (C_d(n)))</th>
<th>(\sum_{n} (T(COSTS)))</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>20</td>
<td>1985</td>
<td>476</td>
<td>23.98</td>
<td>12</td>
<td>13</td>
<td>10.66</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>723</td>
<td>2091</td>
<td>22.82</td>
<td>302</td>
<td>32</td>
<td>12.60</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>40</td>
<td>2223</td>
<td>4923</td>
<td>13.47</td>
<td>694</td>
<td>44</td>
<td>6.34</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>50</td>
<td>8327</td>
<td>148,823</td>
<td>8.54</td>
<td>1599</td>
<td>60</td>
<td>3.75</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>10,091</td>
<td>257,935</td>
<td>21,026</td>
<td>35.4</td>
<td>200</td>
<td>80</td>
<td>3.99</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>10,312</td>
<td>420,056</td>
<td>29,845</td>
<td>3.30</td>
<td>2561</td>
<td>74</td>
<td>2.89</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>38,352</td>
<td>1,231,365</td>
<td>61,457</td>
<td>11.30</td>
<td>2423</td>
<td>108</td>
<td>1.99</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>51,187</td>
<td>1,789,607</td>
<td>81,036</td>
<td>12.10</td>
<td>5912</td>
<td>122</td>
<td>2.06</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>70</td>
<td>102,062</td>
<td>2,550,353</td>
<td>1.38</td>
<td>8307</td>
<td>137</td>
<td>1.65</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>531</td>
<td>6192</td>
<td>2061</td>
<td>32.28</td>
<td>493</td>
<td>83</td>
<td>16.83</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>3183</td>
<td>49,735</td>
<td>13,150</td>
<td>26.44</td>
<td>2230</td>
<td>320</td>
<td>14.34</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>14,409</td>
<td>283,811</td>
<td>44,191</td>
<td>15.57</td>
<td>7826</td>
<td>774</td>
<td>9.89</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>72,112</td>
<td>1,542,793</td>
<td>153,639</td>
<td>9.95</td>
<td>24,342</td>
<td>1,382</td>
<td>5.54</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1127</td>
<td>15,681</td>
<td>6539</td>
<td>41.70</td>
<td>1819</td>
<td>522</td>
<td>28.69</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10,019</td>
<td>172,238</td>
<td>51,145</td>
<td>29.69</td>
<td>8,830</td>
<td>1917</td>
<td>17.70</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>62,280</td>
<td>1,371,885</td>
<td>319,333</td>
<td>23.27</td>
<td>49,634</td>
<td>8,320</td>
<td>16.76</td>
<td></td>
</tr>
</tbody>
</table>

Let us consider the class of problems with three objectives \((q = 3)\). Then, the worst-case size of the \(G_d(n)\) and COSTS sets grows quadratically with depth and range of costs (and so does the worst-case number of dominance comparisons). However, with t-discarding this growth is limited to a linear case. The next section presents a set of experiments to evaluate the effectiveness of this method in practice.

### 4. Experiments

In this section we analyze the time performance of three different versions of the NAMOA* algorithm. We have carried out experiments with two classes of standard multiobjective problems: randomly generated grids and realistic road maps. All versions of NAMOA* employ the ideal point lower bound proposed by Tung and Chew [38] for all problems. The ideal point of each node \(n\) is a lower bound estimate of the cost of all paths from \(n\) to the goal (see Section 3).

The three versions of NAMOA* differ in the order of selection of OPEN labels and/or in the way dominance is checked in filtering and cl-pruning operations. The first variant is called NAMOA*, and uses a lexicographic order of selection. The second, called NAMOAlin, uses an order of selection based on a linear aggregate of vector components. Both NAMOA* and NAMOAlin use standard dominance checks for filtering and cl-pruning. This is, to the best of our knowledge, the usual technique in previously reported experimental evaluations of MO search algorithms.

The third algorithm analyzed, NAMOA*lin, uses a lexicographic order of selection and the t-discarding technique for filtering and cl-pruning described in Section 3.

Notice that all three versions of NAMOA* are exact algorithms, i.e., given enough time and space, they find the set of all non-dominated solution costs \((C^*)\) and the set of all non-dominated solution paths to any problem. All the considered versions of NAMOA* expand the same set of labels. This set is determined solely by the quality of the lower bound estimate, as long as the non-dominated label is always selected for expansion (see Theorem 5.9 [26]). In other words, they all generate the same set of permanent labels \(G_d(n)\) for all explored nodes, whose overall size \(\sum_{n} (G_d(n))\) equals the number of iterations of the algorithms. Additionally, NAMOA*lin and NAMOA*lin (which employ the same selection order) scan the labels exactly in the same order. Thus, the different time performance of these versions can only be attributed to the number and dimensionality of the pruning and filtering operations performed.
The algorithms were implemented to share as much code as possible. The programming language used was ANSI Common Lisp in the LispWorks Professional 6.01 (64-bit) environment. The OPEN queue of alternatives is implemented as a binary heap but only the current best estimate of each node is kept in OPEN at each iteration. The $G_{op}$ and $G_{cl}$ sets are ordered lists according to the label selection policy employed by the algorithm.

Grid problems were run on an Intel Core i7 3612QM at 2.1 GHz and 4 GB of DDR3 RAM, while road map problems were run on a Sun Fire X4140 server with 2 six-core AMD Opteron 2435 @ 2.60 GHz processors and 64 GB of DDR2 RAM, due to their greater memory space requirements. All experiments were run on a single thread.

4.1. Random grid problems

In the first set of experiments, we use randomly generated grids which are a standard testbed in the evaluation of multicriteria search algorithms [21,31]. In particular, we generated square bidimensional grids of $100 \times 100$ nodes with a vicinity of four neighbors. The start node is placed at the grid center (50,50). A single goal node is placed in the diagonal from the center to the bottom right corner. Different solution depths are considered, varying from 20 to 100, i.e. for solution depth $d$, the goal node is at coordinates $(50 + d/2, 50 + d/2)$. A set of five different problems was generated for each solution depth. All displayed data in figures and tables for each solution depth are the averages for those 5 problems. For each arc, $q$ integer scalar costs

![Graph](https://via.placeholder.com/150)

Fig. 2. Average percentage of pruned and filtered labels over the total number of discarded labels by NAMOA* per solution depth for $q \in \{3, 4, 5\}$ objectives in grid problems.
$c(i,j) = (c_1, c_2, \ldots, c_q)$ were randomly generated in the range [1,10] using an uniform distribution, i.e. leading to uncorrelated objectives.

All problem instances were exactly solved by the three algorithms. Table 2 shows the size of relevant sets of labels for the execution of NAMOA\textsuperscript{n}dr on random grid problems. The first column ($q$) indicates the number of objectives. The second column ($d$) displays solution depth. Columns 3, 4, 5, 7 and 8 show values of different magnitudes for all problem instances solved with $q$ objectives at solution depth $d$ averaged over 5 different problem instances. The third column (Max OPEN) displays the average maximum cardinality of the set of open labels. The fourth column, $P_{nj}(Gcl(n))$, displays the total number of closed labels at termination (calculated as the sum of the number of labels at the $Gcl$ sets of all visited nodes), averaged for the 5 problem instances. This is also the average number of labels expanded by the algorithm for these instances. The fifth column shows the sum of sizes of the corresponding sets of truncated labels, i.e. $\sum_{\text{size}}(T(Gcl(n)))$ averaged for the 5 instances. The sixth column displays the percentage ratio between columns four and five. For example, for $d=20$ and $q=3$, the average number of label expansions by NAMOA\textsuperscript{n}dr was 1985, while the average number of labels in the truncated sets was only 476, this results in a percentage ratio of 23.98%.

The seventh column in Table 2 shows the number of different non-dominated solution vectors in COSTS at termination, averaged for the 5 problem instances, i.e. $\vert S \vert$. The eighth column displays the average size of the corresponding truncated set $\vert T(C^*) \vert$, i.e. the one used by NAMOA\textsuperscript{n}dr for filtering operations. The last column displays the percentage ratio between columns seven and eight.

Table 3 displays the runtimes of the three versions of NAMOA\textsuperscript{n} with $q$ objectives, $q \in \{3, 4, 5\}$, for random grid problems.
Values are once again averaged over all 5 problems instances tested for each pair \((q, d)\) of values. The last columns show the relative percentage improvement of NAMOA* \(_d\) over NAMOA* \(_o\) and NAMOA* \(_s\), respectively.

All three evaluated algorithms perform op-pruning in the same way. However, NAMOA* \(_s\) uses a different technique for cl-pruning and filtering.

Fig. 2 displays some results of the execution of NAMOA* \(_d\): the percentage of labels pruned by \(G_{op}\) (op-pruning), truncated closed node labels \(T(G_d)\) (cl-pruning), and filtered by \(T(C^*)\) over the total number of discarded labels. Results are displayed as a function of solution depth \(d\), \((a)\) for \(q = 3\) objectives, \((b)\) for \(q = 4\) objectives, and \((c)\) for \(q = 5\) objectives.

Fig. 3 shows average runtimes of NAMOA* \(_d\), NAMOA* \(_o\), and NAMOA* \(_s\) in logarithmic scale against solution depth for \(q = (3, 4, 5)\) objectives. The items in the legend indicate the version of the algorithm and the objectives, e.g. \(d3\) indicates NAMOA* \(_d\) with \(q = 4\). Finally, Fig. 4 displays the percentage of runtimes of NAMOA* \(_d\) and NAMOA* \(_o\) over NAMOA* \(_s\) for \(q = 3\) as a function of solution depth.

4.2. Road map problems

The second set of experiments involves the use of two realistic road maps. One was obtained from the “9th DIMACS Implementation Challenge: Shortest Path”. The challenge comprises a set of twelve road maps of increasing size.\(^2\) In particular, we use the New York (NY) city map. Additionally, we have employed a second road map from the US Census 2000 TIGER/Line Files, which was not directly used for the challenge but is available from the same site. In particular, we use the Vermont State (VT) map.

All DIMACS maps provide two different criteria: physical distance and travel time. An economic cost criterion was further introduced in [20]. This was obtained combining tolls and fuel consumption according to road category. The resulting values are not linearly correlated to those of the other criteria. The experiments reported below consider the simultaneous minimization of these three attributes, physical distance \((c_1)\), travel time \((c_2)\), and economic cost \((c_3)\). Only algorithms NAMOA* \(_d\) and NAMOA* \(_o\) are tested over road maps, since NAMOA* \(_s\) clearly outperforms NAMOA* \(_s\).

The two maps selected for our experiments have significantly different sizes. Table 4 shows the coordinates and number of nodes and arcs for each map. Renderings of the NY and VT cut maps are presented in Figs. 5 and 6, respectively. The VT cut map corresponds to a trimmed version of the original map of Vermont, reduced to approximately 70% of its original size. The size of this map allows the solution of the complete set of experiments by both algorithms in reasonable time. We use the test set on this map to compare the relative performance of the two versions of NAMOA* over a realistic problem in a similar way as we did over random grids in Section 4.1.

The NY city map is considerably larger. Previously reported runtimes of NAMOA* solving a set of random biobjective problems on this map (minimizing \(c_2\) and \(c_3\)) were up to several days [20]. Since we deal with three-objective problems, which require more computational effort, we establish a runtime limit of 8 h for each problem instance. We use the test set on this map to analyze the range of problems that can be exactly solved in practical time by both versions of NAMOA*.

This procedure has been previously used to compare exact algorithms on difficult road map problem instances [31,32,22].

For the VT cut map we generated a set of twenty problems where start and goal nodes were randomly chosen using a uniform distribution. These test sets and the additional map files are available online.\(^3\)

Both NAMOA* \(_d\) and NAMOA* \(_o\) exactly solved all twenty problem instances. Table 5 displays the size of relevant label sets for each problem instance solved by NAMOA* \(_d\), as well as runtimes for NAMOA* \(_o\) and NAMOA* \(_s\). The first column displays the problem identifier \((n)\). The description of these sets is the same presented in Section 4.1 for Table 2. Fig. 7 shows the runtimes of NAMOA* \(_d\) and NAMOA* \(_o\) for the VT cut map sorted by the number of labels expanded by each problem.

Regarding the New York city road map, we selected the first twenty problems proposed in [18]. These were randomly generated using a uniform distribution to select start and goal nodes. Table 6 displays the size of relevant label sets for each problem instance solved by NAMOA* \(_d\), as well as runtimes of NAMOA* \(_o\) and NAMOA* \(_s\). Notice that for the NY map the algorithms were not capable of solving several problem instances in the given 8 h time limit. These are indicated by symbol “−” in the table. Problems #11 and #6 were solved by NAMOA* \(_s\) without time limit in 31 h and 25 days, respectively.

Finally, Figs. 8 and 9 show the percentage of labels filtered, pruned by open, and pruned by closed node labels over the total number of discarded labels by NAMOA* \(_o\). The horizontal axis shows problem identifiers sorted by the number of expanded labels.

5. Discussion

To the best of our knowledge, this paper reports the largest three-objective search problems over road maps solved to date. Regarding the algorithms using the standard dominance checks (NAMOA* \(_o\) and NAMOA* \(_s\)), our experiments on random grid problems confirm that a linear aggregate selection function is consistently more efficient than a lexicographic one on grid problems with three, four and five objectives. This is in accordance with recently reported results over biobjective problems [15,21]. However, using a lexicographic order allows efficient dominance checks when combined with r-discarding for filtering and cl-pruning (NAMOA* \(_s\)). Results over grid problems reveal a dramatic improvement in time performance of over an order of magnitude for three-objectives problems (see Fig. 3). The speedup of NAMOA* \(_o\) over NAMOA* \(_s\) even grows with problem difficulty (see Table 3 and Fig. 4), reducing time requirements over 90% for the harder three-objective problems. When more objectives are considered, \(q = (4, 5)\), similar results can be observed, although a smaller number of experiments can be presented due to the increasing difficulty.

As expected, multiobjective label-setting search spends most of the time performing dominance checks between labels. Every new
Table 5
Size of relevant sets of labels for VT\(_\text{cut}\) road map problems solved by NAMOA\(_n\). Magnitudes \(\sum_{i} |G_{i}(n)|\) and \(|C^n|\) are the same for NAMOA\(_n\).

<table>
<thead>
<tr>
<th>n</th>
<th>Max OPEN</th>
<th>Size of relevant sets of labels of NAMOA(_n)</th>
<th>Size of relevant sets of labels of NAMOA(_n)dr</th>
<th>Runtime (s)</th>
<th>(t_{\text{NAMOA}_n})</th>
<th>(t_{\text{NAMOA}_n\text{dr}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1352</td>
<td>209,906</td>
<td>27,353</td>
<td>10.03</td>
<td>40.0</td>
<td>9.2</td>
</tr>
<tr>
<td>2</td>
<td>2157</td>
<td>114,109</td>
<td>40,522</td>
<td>35.51</td>
<td>7.3</td>
<td>4.6</td>
</tr>
<tr>
<td>3</td>
<td>1062</td>
<td>11,483</td>
<td>11,477</td>
<td>99.95</td>
<td>0.29</td>
<td>0.45</td>
</tr>
<tr>
<td>4</td>
<td>2717</td>
<td>1,132,450</td>
<td>59,068</td>
<td>5.22</td>
<td>1428.4</td>
<td>47.5</td>
</tr>
<tr>
<td>5</td>
<td>922</td>
<td>44,950</td>
<td>10,398</td>
<td>23.13</td>
<td>2.6</td>
<td>2.0</td>
</tr>
<tr>
<td>6</td>
<td>4373</td>
<td>5,445,252</td>
<td>160,410</td>
<td>2.95</td>
<td>17913.2</td>
<td>259.0</td>
</tr>
<tr>
<td>7</td>
<td>61</td>
<td>178</td>
<td>178</td>
<td>100</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>8</td>
<td>62</td>
<td>483</td>
<td>483</td>
<td>100</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>9</td>
<td>1636</td>
<td>64,225</td>
<td>43,787</td>
<td>68.18</td>
<td>3.4</td>
<td>2.4</td>
</tr>
<tr>
<td>10</td>
<td>7271</td>
<td>5,229,959</td>
<td>32,398</td>
<td>6.2</td>
<td>17167.5</td>
<td>246.0</td>
</tr>
<tr>
<td>11</td>
<td>31,846</td>
<td>10,057,176</td>
<td>286,083</td>
<td>2.84</td>
<td>28712.6</td>
<td>527.5</td>
</tr>
<tr>
<td>12</td>
<td>1974</td>
<td>127,731</td>
<td>15,947</td>
<td>12.48</td>
<td>39.2</td>
<td>4.7</td>
</tr>
<tr>
<td>13</td>
<td>9387</td>
<td>8,640,728</td>
<td>137,410</td>
<td>1.59</td>
<td>27556.0</td>
<td>395.2</td>
</tr>
<tr>
<td>14</td>
<td>819</td>
<td>46,861</td>
<td>9650</td>
<td>20.59</td>
<td>3.8</td>
<td>1.8</td>
</tr>
<tr>
<td>15</td>
<td>12,648</td>
<td>568,338</td>
<td>90,676</td>
<td>15.95</td>
<td>108.6</td>
<td>33.3</td>
</tr>
<tr>
<td>16</td>
<td>1558</td>
<td>87,522</td>
<td>51,641</td>
<td>59.00</td>
<td>7.6</td>
<td>4.7</td>
</tr>
<tr>
<td>17</td>
<td>2596</td>
<td>1,207,119</td>
<td>41,414</td>
<td>3.43</td>
<td>20777.7</td>
<td>51.2</td>
</tr>
<tr>
<td>18</td>
<td>1331</td>
<td>10,270</td>
<td>5550</td>
<td>5.04</td>
<td>0.31</td>
<td>0.42</td>
</tr>
<tr>
<td>19</td>
<td>34</td>
<td>92</td>
<td>92</td>
<td>100</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>20</td>
<td>24,671</td>
<td>1,856,420</td>
<td>171,675</td>
<td>9.25</td>
<td>16999.9</td>
<td>102.5</td>
</tr>
</tbody>
</table>

Fig. 7. Runtimes of NAMOA\(_n\) and NAMOA\(_n\)dr for the VT\(_\text{cut}\) map problems sorted by the number of labels expanded.

Table 6
Results for NY road map problems with size of relevant sets of labels of NAMOA\(_n\) and runtimes of NAMOA\(_n\) and NAMOA\(_n\)dr. Magnitudes \(\sum_{i} |G_{i}(n)|\) and \(|C^n|\) are the same for NAMOA\(_n\).

<table>
<thead>
<tr>
<th>n</th>
<th>Max OPEN</th>
<th>Size of relevant sets of labels of NAMOA(_n)</th>
<th>Size of relevant sets of labels of NAMOA(_n)dr</th>
<th>Runtime (s)</th>
<th>(t_{\text{NAMOA}_n})</th>
<th>(t_{\text{NAMOA}_n\text{dr}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>379,060</td>
<td>274,567,814</td>
<td>2,158,829</td>
<td>0.79</td>
<td>45</td>
<td>0.05</td>
</tr>
<tr>
<td>2</td>
<td>185</td>
<td>17,294</td>
<td>1,771</td>
<td>10.24</td>
<td>1.4</td>
<td>0.7</td>
</tr>
<tr>
<td>3</td>
<td>–</td>
<td>–</td>
<td>1.26</td>
<td>12</td>
<td>3.96</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>8636</td>
<td>3,390,656</td>
<td>28,088</td>
<td>0.83</td>
<td>3963.9</td>
<td>149.3</td>
</tr>
<tr>
<td>5</td>
<td>44</td>
<td>719</td>
<td>613</td>
<td>85.26</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>6</td>
<td>152,988</td>
<td>80,721,099</td>
<td>628,829</td>
<td>0.78</td>
<td>11,641.6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>160,079</td>
<td>182,473,300</td>
<td>1,118,218</td>
<td>0.61</td>
<td>21,768.3</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>464,998</td>
<td>214,901,344</td>
<td>1,070,285</td>
<td>0.50</td>
<td>26,078.7</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>35,544</td>
<td>24,584,323</td>
<td>812,383</td>
<td>3.30</td>
<td>1,452.5</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>159,041</td>
<td>278,481,469</td>
<td>6,296,377</td>
<td>2.26</td>
<td>–</td>
<td>21957.6</td>
</tr>
<tr>
<td>15</td>
<td>844,037</td>
<td>136,776,273</td>
<td>5,256,283</td>
<td>3.84</td>
<td>16,582.9</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>6821</td>
<td>2,445,191</td>
<td>242,832</td>
<td>9.93</td>
<td>765.5</td>
<td>106.8</td>
</tr>
<tr>
<td>17</td>
<td>16</td>
<td>236,826</td>
<td>270,364,947</td>
<td>1.63</td>
<td>25,590.0</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>67,883</td>
<td>108,347,749</td>
<td>1,137,035</td>
<td>1.05</td>
<td>8335.3</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>482,686</td>
<td>162,419,342</td>
<td>1,788,698</td>
<td>1.10</td>
<td>15,617.3</td>
<td></td>
</tr>
</tbody>
</table>
label has to be checked for op-pruning, cl-pruning, and filtering. Fig. 2 shows that, for the harder grid problems, cl-pruning is the operation that tends to discard most labels for deeper solution depths. The same tendency can be observed regardless of the number of objectives.

While the ratio of labels discarded by filtering decreases with problem difficulty, it was always larger than the ratio of those discarded by op-pruning in our grid experiments. This is important for the efficiency of NAMOA$_{dr}^*$, since cl-pruning and filtering can both benefit from $t$-discarding. Table 2 reveals that the truncated sets of labels used by $t$-discarding are significantly smaller than the original ones, and their relative size even decreases with problem difficulty. For example, with solution depth $d=100$ and $q=3$, a label is checked against a set of 8307 labels for discarding in the worst case involving only a set of 137 labels (or 1.65 %).

These results are confirmed by the experiments on the realistic road map problems. These are much harder problems. The hardest one solved (14 of NY) required up to 278 million label expansions (the most difficult grid problems with $q=3$ involved in average 2.5 million label expansions, and 3.3 million in the worst case). Again, NAMOA$_{dr}^*$ clearly outperformed NAMOA$_{lin}^*$, which could only solve problems involving less than 10.1 million label expansions (11 of VT$_{cut}$). For such a problem, (11 of VT$_{cut}$), NAMOA$_{dr}^*$ required only 1.83% of the time needed by NAMOA$_{lin}^*$. In fact, we solved without time limit the two smallest problems of the NY map not solved by NAMOA$_{lin}^*$ within the time limit. Problem #11 was solved in 31 h, and problem #6 was solved in 25 days.

The test set for the VT$_{cut}$ map was entirely solved by both algorithms. NAMOA$_{dr}^*$ required 1.74% of the time needed by NAMOA$_{lin}^*$ to solve the complete set of problems (see Table 7). Regarding the NY city test set, NAMOA$_{lin}^*$ was capable of solving only 4 problems out of 20 within the time limit (20%), while NAMOA$_{dr}^*$ solved 14 of them (70%) (see Table 6). Therefore, NAMOA$_{dr}^*$ not only outperforms NAMOA$_{lin}^*$ in time efficiency, but also extends the range of problems that can be exactly solved within the given time limit. The largest road map problem solved by NAMOA$_{dr}^*$ comprises 10,057,176 labels (11 of VT$_{cut}$), while the largest one solved by NAMOA$_{lin}^*$ comprises 278,481,469 labels (14 of NY).

Except for the simpler problems, cl-pruning was responsible for around 70–80% of the discarded labels (see Figs. 8 and 9). In general, the ratio of filtered labels was larger than those discarded by op-pruning. Once again, this explains the efficiency achieved by $t$-discarding. Tables 5 and 6 show dramatic reductions in the sizes of the sets used for cl-pruning and filtering. For the hardest solved instance (14 of NY), the size of $T(C^*)$ is just 0.32% the size of $C^*$.

In summary, the proposed $t$-discarding procedure shows to be very effective, reducing the time requirements over an order of magnitude when compared with the most efficient search using the standard procedure of dominance checking. The new technique
effectively extends the size of the many-objective problems that can be practically solved.

6. Conclusions and future work

This paper develops and tests a new dimensionality reduction technique for dominance checks in multiobjective exact label-setting algorithms. The technique, t-discarding, requires a lexicographic order for label selection, and a monotone lower bound function. With t-discarding, cl-pruning and filtering can be performed more efficiently. In this work we applied t-discarding to NAMOA*.

All three analyzed versions of NAMOA* (NAMOA*, NAMOA*ex, and NAMOA*dr) are exact algorithms, i.e. given enough computational space and time they find the set of all non-dominated solution paths to the problem. They all scan the same sets of labels and, in particular, they visit the same set of nodes, and generate the same sets of permanent labels Cdr for them. In the case of NAMOA*ex and NAMOA*dr, they even explore the same labels in the same order. However, while standard NAMOA* performs filtering and cl-pruning against the costs and Cdr sets respectively, the t-discarding version of NAMOA* (NAMOA*tdr) performs these checks much more efficiently against the smaller sets T(COSTS) and T(Cdr).

Experimental results over one-to-one random grids and road map problems show a dramatic reduction in time requirements for all search problems, regardless of the number of objects considered. For the random grid problems we report the performance of all algorithms. In this case, NAMOA*ex reduces time requirements over an order of magnitude. Similar results are obtained for the VRP, road map problems. In the case of the NY city map problems we observed that NAMOA*ex is capable of solving exactly much larger problem instances than NAMOA*dr within a given 8 h time limit. Using this limit, NAMOA*ex was capable of solving a problem requiring the examination of over 278 million labels, while the hardest road map problem instance solved by NAMOA*dr required the examination of slightly over 10 million nodes. Removing the time limit, NAMOA*ex solved a problem requiring around 80 million labels in 25 days, while NAMOA*dr required only 3.23 h for the same problem.

Regarding future work, the same scheme could be applied to one-to-all multiobjective search problems as well. In this case, filtering is not needed. However, the results presented here suggest that much of the time gain results from efficient cl-pruning.

Another interesting avenue of future research is the combination of vector frontier search with lower bound distance estimates. This algorithm uses a similar dimensionality reduction technique to reduce space requirements. However, it requires an additional path-recovery phase that requires additional time over NAMOA*.

The use of t-discarding could reduce this extra effort.

References