An evaluation of best compromise search in graphs [⋆]

Enrique Machuca[1], Lawrence Mandow[1], and Lucie Galand[2]

[1] Univ. Malaga (Spain), {machuca, lawrence}@lcc.uma.es
[2] Univ. Paris-Dauphine (France), lucie.galand@dauphine.fr

Abstract. This work evaluates two different approaches for multicriteria graph search problems using compromise preferences. This approach focuses search on a single solution that represents a balanced tradeoff between objectives, rather than on the whole set of Pareto optimal solutions. We review the main concepts underlying compromise preferences, and two main approaches proposed for their solution in heuristic graph problems: naive Pareto search (NAMOA*), and a k-shortest-path approach (kA*). The performance of both approaches is evaluated on sets of standard bicriterion road map problems. The experiments reveal that the k-shortest-path approach looses effectiveness in favor of naive Pareto search as graph size increases. The reasons for this behavior are analyzed and discussed.

1 Introduction

Multicriteria optimization problems involve the consideration of different objectives that need to be optimized simultaneously. These problems seldom have a single optimal solution, and in general, many optimal trade-offs between the different objectives can be considered. The set of *rational* decisions to the problem is defined by the set of non-dominated (*Pareto-optimal*) solutions. A nondominated solution cannot be improved by other solution in one objective without worsening in at least another one.

However, choosing one among the set of nondominated solutions is a subjective decision particular to each decision maker. Several approaches have been proposed to tackle this question. These include goal satisfaction, multiattribute utility theory, or compromise programming [1]. One of the most popular approaches in multicriteria decision making is based on the use of *achievement scalarizing functions*. These functions evaluate a solution according to its *distance* to a reference point in the objective space [12]. The most preferred solution is then the closest one to the reference point. Such a solution is called hereafter a *best compromise solution*. The reference point can be specified by the decision maker as her aspiration level on each objective [10]. Otherwise, the reference point can be the *ideal point*, defined as the cost of an ideal (but generally unreachable) solution that would achieve the scalar optimal value for all objectives [14].

In this paper, we consider the problem of determining a *best compromise* path from an initial node to a goal node in a graph where the arcs are valued by several objective functions. In this setting, the value of a path is the componentwise sum of the value of its arcs. Since a best compromise solution is Pareto optimal (for any rational decision maker), a simple approach could be a two-step procedure: 1) generate the set of Pareto optimal paths, and 2) select the best compromise path among them. Efficient algorithms have been proposed to generate the whole set of Pareto optimal paths [7]. Contrary to the single objective case, several distinct paths can be (Pareto) optimal on a node. The number of Pareto optimal paths can even be exponential in the size of the instance [4].

[⋆] The final publication is available at http://link.springer.com/chapter/10.1007/978-3-642-40643-0_1

Pareto dominance tests have to be performed to compare subpaths reaching each node, requiring significant computation times. More specific approaches have been proposed to directly focus the search on a preferred path with respect to a specific preference model without generating the whole set of Pareto optimal solutions ([8,2,3]). Among these, kA* relies on an ordered enumeration of the paths (k*-shortest-path* algorithm) according to a linear scalarizing function, until one obtains the guarantee that the best compromise path has been found (i.e. already enumerated). Thanks to the use of a linear scalarization, this approach does not require costly Pareto dominance tests.

In this paper, we compare two different heuristic algorithmic strategies for best compromise search in multiobjective graphs. The first one is a two-step procedure based on standard Pareto search algorithms, like NAMOA* [7]. The second one explores the use of k-shortest-path algorithms to avoid evaluating all Pareto optimal paths in the graph, as well as performing computationally costly Pareto dominance tests. Preliminary results [2] showed an advantage in performance for kA* against the naive Pareto search (NAMOA*) on random graphs with an artificially calculated heuristic. This paper performs a more systematic evaluation of both approaches on sets of standard bicriterion route planning problems [9,6,5]. The algorithms are provided with the precalculated Tung-Chew heuristics [11]. The experiments reveal that the k-shortest-path approach looses effectiveness in favor of naive Pareto search as graph size increases.

Section 2 reviews relevant concepts and previous work that are used in this paper. After the presentation of the instances in Section 3, the results obtained are shown in Section 4. Then they are analyzed in Section 5 to exhibit the advantages and the drawbacks of the two approaches. Finally, some conclusions are summarized in Section 6, leading us to further research perspectives.

2 Related work

2.1 Preliminaries

Let us consider a decision problem where X denotes the set of feasible alternatives and each feasible alternative $x \in X$ is evaluated according to a set of q objective functions to be minimized $f_i : X \to \mathbb{R},\ i \in \{1..q\}$. Each alternative x is represented in the objective space by an evaluation vector (vector cost) $\boldsymbol{f}(x) = (f_1(x), \ldots, f_q(x))$. Let $Y = f(X)$ denote the set of images of the feasible alternatives of X in the objective space. The comparison of the elements of X boils down to the comparison of their vector costs in Y. Let us define the dominance relation ($\prec$) between vectors as follows,

$$\forall \boldsymbol{y}, \boldsymbol{y}' \in \mathbb{R}^q \quad \boldsymbol{y} \prec \boldsymbol{y}' \quad \Leftrightarrow \quad \forall i \quad y_i \leq y'_i \ \wedge \ \boldsymbol{y} \neq \boldsymbol{y}' \tag{1}$$

where y_i denotes the i-th element of vector $\boldsymbol{y}$.

Given a set of vectors Y, we shall define $\mathcal{N}(Y)$ the set of non-dominated (Pareto-optimal) vectors in set Y in the following way,

$$\mathcal{N}(Y) = \{\boldsymbol{y} \in Y \mid \nexists \boldsymbol{y}' \in Y \quad \boldsymbol{y}' \prec \boldsymbol{y}\} \tag{2}$$

The set $\mathcal{N}(Y)$ is bounded by the *ideal point* $\boldsymbol{\alpha} = (\alpha_1 \ldots \alpha_q)$ and the *nadir point* $\boldsymbol{\beta} = (\beta_1 \ldots \beta_q)$, where $\alpha_i = min_{\boldsymbol{y} \in \mathcal{N}(Y)}\{y_i\}$ and $\beta_i = max_{\boldsymbol{y} \in \mathcal{N}(Y)}\{y_i\}$ [3].

[3] For $q = 2$ these points are easily obtained by the heuristic precalculation procedure [11]

2.2 Scalarizing functions

Resorting to scalarizing functions amounts to modifying the multiobjective optimization problem into a single objective one. Preferential information is taken into account through parameters used in the scalarizing functions (e.g. weights to define the importance of the objectives). An adequate scalarizing function s is required to have the following properties: (1) any non-dominated solution can be optimal with respect to s (with an appropriate choice of parameters); and (2) any optimal solution with respect to s has to be non-dominated. These requirements ensure that, for any rational decision maker, the preferred solution can be reached optimizing some scalarizing function.

One of the simplest multicriteria approaches is to define the scalarizing function as a linear weighted combination of the evaluation vector. Given a set of weights w_i, the goodness or *utility* of a solution is given by,

$$u(\boldsymbol{y}) = \sum_i w_i \; y_i \tag{3}$$

Any solution minimizing $u(\boldsymbol{y})$ will be non-dominated (second requirement). However, in general only a subset of all non-dominated solutions can be obtained (the so-called *supported* solutions). Some non-dominated solutions cannot be obtained regardless of the chosen weights. The first requirement is thus not satisfied.

Achievement scalarizing functions are widely used in multicriteria decision making. They estimate the distance of a solution to a reference point using Minkowski's distance, or ℓ_p-norm, defined by:

$$\ell_p(\boldsymbol{y}) = \|\boldsymbol{y}\|_p = (\sum_i |y_i|^p)^{1/p} \qquad (p \geq 1) \tag{4}$$

Different norms are obtained for different values of p. The case for $p = 1$ is called *Manhattan* distance, $p = 2$ is the *Euclidean* distance, and $p = \infty$ is the *Chebyshev* distance, which measures the maximum component.

In the absence of further preferential information, and without loss of generality, we may consider that the preferred solution is the one that minimizes distance to the ideal point. For example, the Manhattan and Chebyshev distances from a vector $\boldsymbol{y}$ to the ideal point $\boldsymbol{\alpha}$ are defined respectively as,

$$\|\boldsymbol{y} - \boldsymbol{\alpha}\|_1 = \sum_i |y_i - \alpha_i| \tag{5}$$

$$\|\boldsymbol{y} - \boldsymbol{\alpha}\|_\infty = \max_i |y_i - \alpha_i| \tag{6}$$

Notice that any solution that minimizes ℓ_p distance to the ideal point for some p is a non-dominated solution (second requirement), except for the case $p = \infty$ [13]. In the latter case, there is at least one ℓ_∞-optimal solution that is non-dominated, but this may dominate other ℓ_∞-optimal ones. However when $p = \infty$, any non-dominated solution can minimize ℓ_p distance. The first requirement is then satisfied. Actually, there does not exist any scalarizing function satisfying simultaneously the two requirements [13]. Chebyshev distance appears thus as an adequate achievement function which enables to reach any potentially preferred solution. We define therefore in the following a *best compromise solution* as a solution that minimizes Chebyshev distance to the ideal point. We use the following scalarizing functions for Manhattan and Chebyshev norms[4],

$$s^1(\boldsymbol{y}) = \sum_i w_i(y_i - \alpha_i) = \sum_i w_i y_i - \sum_i w_i \alpha_i \tag{7}$$

$$s^\infty(\boldsymbol{y}) = \max_i w_i(y_i - \alpha_i) \tag{8}$$

where for all i, $w_i = \frac{\delta_i}{\beta_i - \alpha_i}$ and δ_i is the relative importance of objective i.

2.3 Compromise search with Chebyshev norm

We consider the problem of determining a best compromise path from an initial node to a goal node in a multiobjective graph with respect to the Chebyshev norm. Notice that partial solutions evaluated according to the Chebyshev norm do not satisfy Bellman's optimality principle [2]. We compare two main general approaches. The first (*naive*) approach consists in calculating the set of all nondominated solution costs using a label setting algorithm like NAMOA* [7], and then identifying the optimal solution among them. This approach is simple, but: (a) requires the calculation of the full Pareto set, (b) costly Pareto dominance tests must be performed during the search to compare the current subpaths and keep only the optimal ones. The second approach is an alternative algorithm based on single-objective k-shortest paths search [2]. The major insight is that it is possible to devise a weighted linear function that minorates the Chebyshev distance [2]. For any vector cost $\boldsymbol{y} \in \mathbb{R}^q$, it can indeed be easily shown that,

$$\frac{s^1(\boldsymbol{y})}{q} \leq s^\infty(\boldsymbol{y}) \tag{9}$$

The linearity of the scalarizing function s^1 makes it possible to determine the optimal path with respect to s^1 from optimal (w.r.t. s^1) subpaths (exploiting Bellman's principle). However, the optimal path with respect to s^1 is not necessarily a path which minimizes the Chebyshev norm. The principle of this approach is then to enumerate the k best paths with respect to s^1 until we are sure to have found the optimal path with respect to s^∞. The procedure can be summarized as follows,

1. Use a k-shortest paths algorithm to generate a sequence of solutions according to s^1. Let us denote by $\boldsymbol{y}_n$ the vector cost of the n-th solution found.
2. For each new solution found, calculate its value according to the Chebyshev norm s^∞, keeping the best value found so far, i.e $p^* = min_n s^\infty(\boldsymbol{y}_n)$.
3. Stop searching as soon as a newly found solution $\boldsymbol{y}_m$ satisfies $\frac{s^1(\boldsymbol{y}_m)}{q} > p^*$.
4. Return the solution that achieved the optimal value of p^*. Path p^* is optimal since for any $r > m$, $\frac{s^1(y_r)}{q} \geq \frac{s^1(y_m)}{q} > p^*$ and by (9) we have $s^\infty(y_r) \geq \frac{s^1(y_r)}{q}$.

Since the k-best search is performed on a single objective version of the problem, no Pareto dominance tests are performed during the search. In the following, we evaluate this approach using kA*, a variant of A* that calculates k-shortest paths [2]. More precisely, we improve kA* to avoid cyclic paths, which can obviously never lead to non-dominated solutions. Otherwise, performance would be quite poor in our test sets.

[4] Note that if we are evaluating *solutions* (not partial solutions), then by definition we have $\forall i \; 0 \leq \alpha_i \leq y_i$.

3 Experiments

The algorithms have been tested on different classes of problem sets taken from the multiobjective search literature: bidimensional grids with random costs, and random route planning problems on road maps. In all cases, the algorithms were provided with precalculated heuristic functions as described by Tung and Chew [11]. These are obtained from ideal optimal values for both objectives calculated with reverse scalar searches, which are computationally much less costly than subsequent multicriteria searches [6].

The first test set involves square grids, like those described in [6]. A vicinity of four neighbours is used. Bidimensional costs are random integers in the range $[1, 10]$. Start node is placed at the center of the grid. For a grid of size $d \times d$, the goal is placed at depth d. Depth varies from 10 to 100 in steps of 10 with 10 problems for each size, i.e., there are 100 problem instances. Thus, the total number of nodes and arcs for the largest-sized grids (200 $\times$ 200) is 40000 and 159200 respectively. The average number of Pareto-optimal solution paths for the ten largest problems (200 $\times$ 200) is 124.8.

The second test set is taken from the work of Raith and Ehrgott [9]. This consists of three modified road maps from the '9th DIMACS Challenge on Shortest Paths': Washington DC (DC), Rhode Island (RI), and New Jersey (NJ). These include integer cost values for two different objectives: time and distance. The maps include a Hamiltonian cycle that guarantees all nodes are connected. Nine random problems are defined for each map. The size of the maps and the average number of distinct Pareto-optimal costs are displayed in table 1.

The final test set consists of fifty random problems over the unmodified New York City (NY) map from the DIMACS Challenge presented at [5]. The hardest road map problems tested for the algorithms appear in this problem set, as reflected in the average number of distinct Pareto (see table 1).

In all cases, problem instances were solved with a 1h time limit. The algorithms were implemented in ANSI Common Lisp using LispWorks 6.0 Enterprise 64 bits. The first and third test sets were run on a Sun Fire X4140 server with 2 six-core AMD Opteron 2435 @ 2.60GHz processors and 64 Gb of RAM, under Windows Server 2008 R2 Enterprise (64-bits). In the second test set, the algorithms were run on a Windows 7 64-bit platform, with an Intel Core2 Quad Q9550 at 2.8Ghz, and 4Gb of RAM.

4 Results

Regarding the grid test set, figure 1 shows average execution times as a function of solution depth (averaged for the ten random problems available for each depth). NAMOA* displays a steady growth of time requirements. In general kA* provides better results except for the hardest problems. Table 2 details the execution times for each of the ten problems at depth 200. Most of the problems are solved rather quickly. However, two of them have very large time requirements referred to their actual nondominated solutions.

Regarding the modified road map test sets (DC, RI, and NJ) maps, each problem was solved in ten different runs. Figure 2 shows average execution times in logarithmic scale. Problem instances in the abscissa axis are ordered by increasing value of ordinate for NAMOA*. Values not displayed exceeded 1h time limit. In the NY City map, NAMOA* was able to solve all 50 instances, against only 15 of them by kA* under the time constraints. Algorithm kA* was faster than NAMOA*only in 3 of them (Fig. 4).

Name	Location	Nodes	Arcs	Avg. nondom. costs
DC	Washington D.C.	9,559	39,377	3.33
RI	Rhode Island	53,658	192,084	9.44
NJ	New Jersey	330,386	1,202,458	10.66
NY	New York City	264,346	730,100	198.62

Table 1. Size of road maps used in the test sets, and average number of nondominated solution costs (DC, RI, and NJ as taken from [9], NY as taken from [5]) .

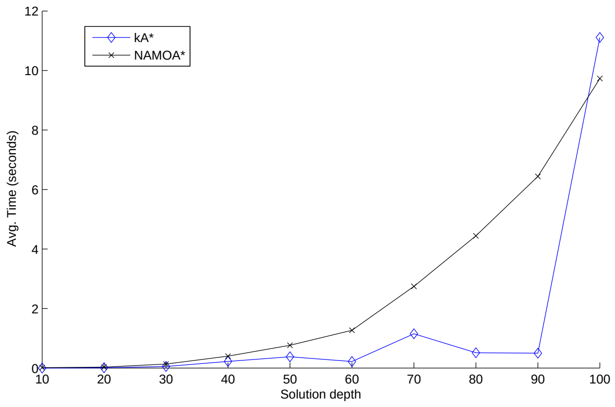


Fig. 1. Average execution times for the square grid problem set, as a function of goal depth.

5 Discussion

The execution time of kA* is much less predictable than that of NAMOA*. This is quite evident from the grid data set, where solution depth can be easily controlled for experimental purposes. In the ten 200×200 grid instances, differences of three orders of magnitude in execution time can be observed for different problems. Table 2 provides valuable information regarding the performance of kA*. For example, in the hardest instances (#3 and #8), the number k of solution paths examined grows to 3280 and 2992 respectively. The number of distinct *solution* vector costs found was 164 and 154. Therefore, the average number of solution paths for each distinct vector cost raises to 20, which makes the algorithm much less competitive in these instances. The table also shows the number of nondominated solution vector costs among those explored by kA*.

Problem	Time (sec.)	k	Sol. vector costs	Avg. paths per sol. cost	Nondom sol. vector costs
1	0.1560	164	81	2.02	23
2	1.0140	382	101	3.78	41
3	57.6270	3280	168	19.52	44
4	0.0940	40	24	1.66	16
5	0.5460	101	36	2.80	16
6	0.8890	530	141	3.75	37
7	0.7800	160	51	3.13	37
8	43.0090	2992	160	18.70	33
9	0.1720	75	35	2.14	17
10	6.8320	808	65	12.43	27

Table 2. Data of kA*for each problem instance of the grid test set (200×200 grids).

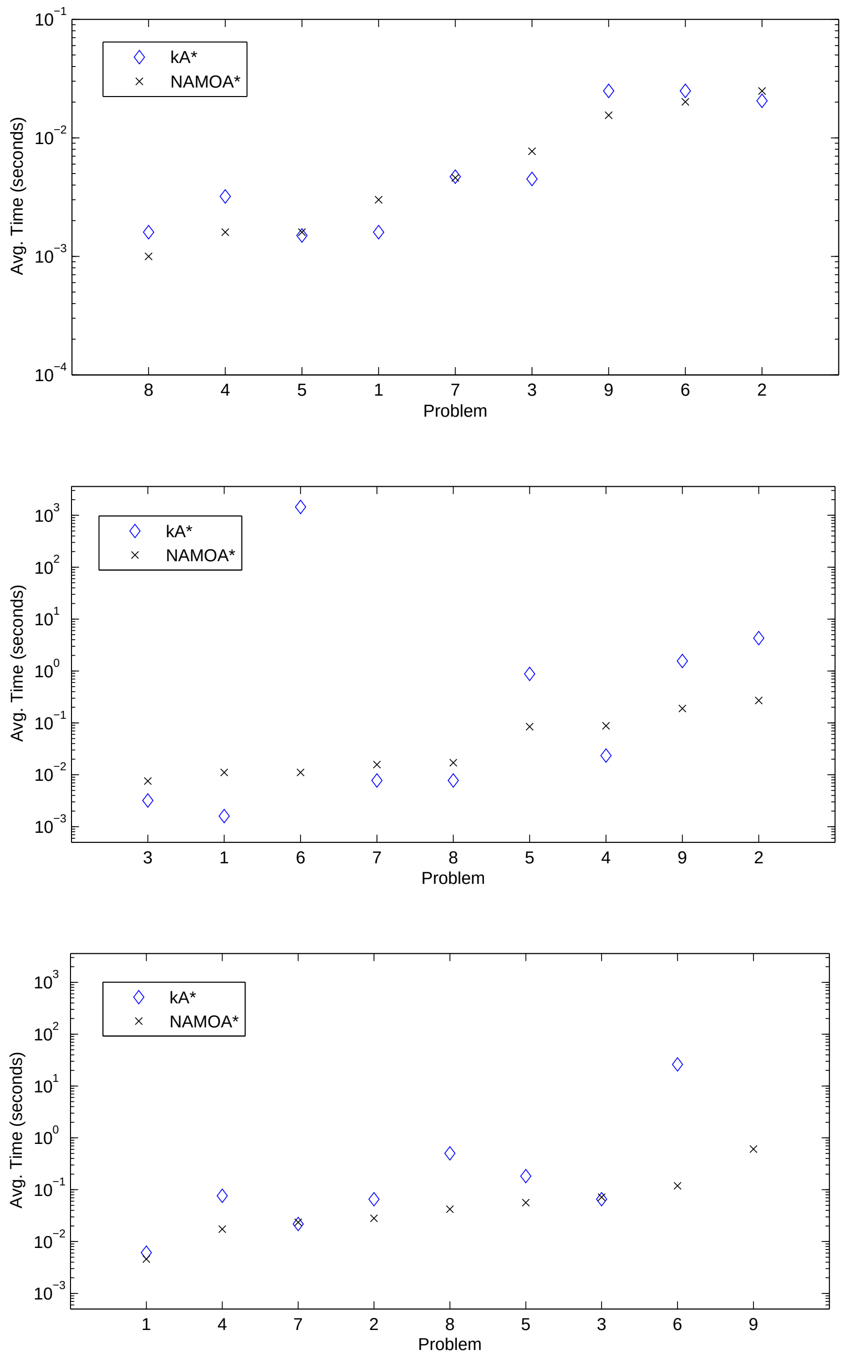
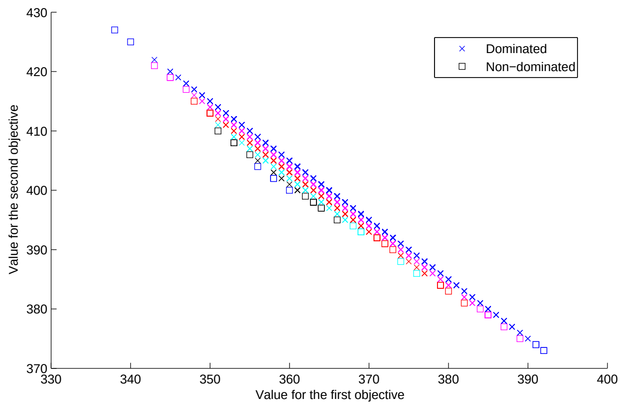
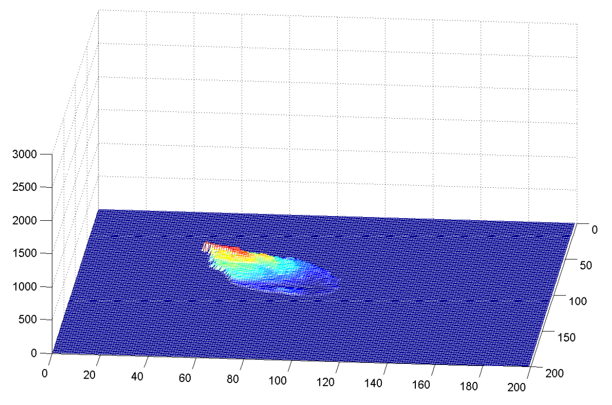


Fig. 2. Time results on modified DIMACS road map problems (DC - top, RI - center, NJ - bottom).





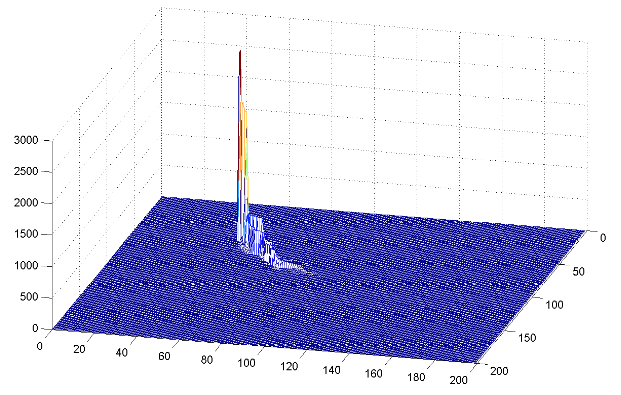


Fig. 3. Analysis of 200 $\times$ 200 grid instance #8: (top) Solution vector costs in cost space; (center) Label expansions in search space (NAMOA*); (bottom) Label expansions in search space (kA*).

The number of labels explored by kA* at each node can increase sharply. This can be attributed to the exponential worst case behavior of the algorithm, i.e. in some problem instances there can be a combinatorially large number of paths with the same vector cost. All of them are explicitly explored by the k-shortest-path approach. In contrast, NAMOA* can be guaranteed to explore the same label for each node only once [7].

Let us analyze instance #8 in more detail. Figure 3(top) displays a portion of cost space with the distinct solution vector costs explored by kA*. Figures 3(center) and 3(bottom) show the number of label expansions for each node in the bidimensional grid for NAMOA*and kA*. NAMOA* explores a larger portion of the grid. The number of nondominated vector costs grows polynomially with depth and, since NAMOA* explores each one only once, the overall number of label expansions is much lower. On the other hand, kA* explicitly explores many different paths with the same vector costs or the same value with respect to s^1 (some of them dominated). The explored portion of the grid is much smaller, and the number of dominated solution vector costs is marginally larger. However, the overall search effort is much higher. This indicates that kA* could be improved if only a single label were explored at each node for each different vector cost, as happens in NAMOA*.

The inability of kA* to solve instance 9 in the NJ map in the given time is also an example of this unpredictability. Results on the NY City map provide further confirmation. Finally, results on road map problems indicate that the performance of kA* is worse than that of NAMOA* in the hardest instances.

6 Conclusions and future work

This paper compares two approaches described in the literature for the calculation of a best compromise path in a multiobjective graph. The first one uses NAMOA*, a multiobjective generalization of A*, to calculate the Pareto set and then determine the best compromise solution. The second one relies on kA*, a k-shortest-paths variant of A*, improved here to avoid cycles. This approach needs to consider dominated paths, but avoids Pareto dominance checks. Earlier tests on small sized problems showed and advantage for the second approach [2]. We perform a more systematic evaluation on a variety of standard problem sets taken from the literature. These comprise random grids as well as realistic route planning problems in road maps. An analysis of the results shows that kA* can indeed be faster in the simpler instances, but looses effectiveness in

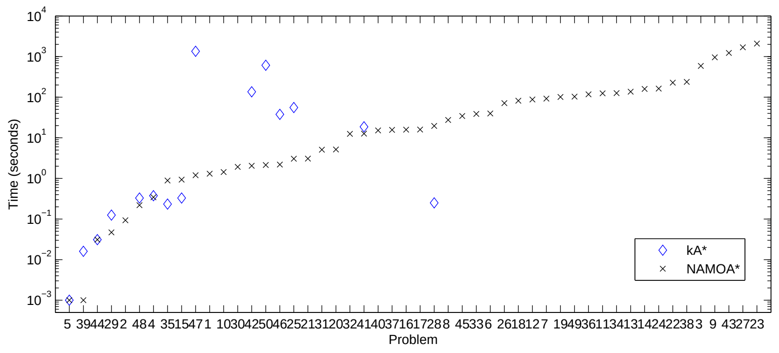


Fig. 4. Time results (in seconds) on the NY City map.

favor of the full Pareto approach as graph size (and hence, problem difficulty) increases. The data also show that the time performance of kA*can be quite unpredictable. This is attributed to the combinatorial nature of the problem, since in some problems there can be a large number of paths with the same vector cost.

Another linear function different to s^1/q could be used to lower bound s^∞. Indeed $1/q$ is not the only weight that could be used. Actually we have $\sum_i \lambda_i w_i(y_i - \alpha_i) \leq s^\infty(y)$ for any nonnegative λ such that $\sum_i \lambda_i = 1$. The number of enumerated solutions depends on the choice of λ. A further study should take this issue into account.

A better understanding on performance of the two very different approaches (kA*and NAMOA*) would make it possible to design a more efficient and more robust algorithm. In particular, cycle avoidance allows effective pruning of some dominated paths on grids and road maps. Other new enhancements of the k-shortest-paths approach should be further investigated. In particular, a variant that only explores the k shortest paths with *different* vector costs is an interesting avenue of future reseach. Our results suggest that a k-shortest-path approach where each label is explored only once for each node, as happens in standard Pareto search, could yield a much more effective algorithm.

Acknowledgments. Partially funded by P07-TIC-03018, Cons. Innovación, Ciencia y Empresa (Junta Andalucía), and Univ. Málaga, Campus Excel. Int. Andalucía Tech.

References

1. V. Chankong and Y.Y Haimes. *Multiobjective Decision Making: Theory and Methodology*. North-Holland, 1983.
2. Lucie Galand and Patrice Perny. Search for compromise solutions in multiobjective state space graphs. In *Proc. of ECAI'2006*, pages 93–97, 2006.
3. Lucie Galand and Olivier Spanjaard. Owa-based search in state space graphs with multiple cost functions. In *FLAIRS Conference 2007*, pages 86–91, 2007.
4. P. Hansen. Bicriterion path problems. In *Lecture Notes in Economics and Mathematical Systems (LNEMS) 177*, pages 109–127. Springer, 1980.
5. E. Machuca and L. Mandow. Multiobjective heuristic search in road maps. *Expert Systems with Applications*, 39:6435–6445, 2012.
6. E. Machuca, L. Mandow, J. L. Pérez-de-la-Cruz, and A. Ruiz-Sepúlveda. A comparison of heuristic best-first algorithms for bicriterion shortest path problems. *European Journal of Operational Research*, 217:44–53, 2012.
7. L. Mandow and J. L. Pérez de la Cruz. Multiobjective A* search with consistent heuristics. *Journal of the ACM*, 57(5):27:1–25, 2010.
8. Patrice Perny and Olivier Spanjaard. On preference-based search in state space graphs. In *Proc. Eighteenth Nat. Conf. on AI*, pages 751–756. AAAI Press, July 2002.
9. Andrea Raith and Matthias Ehrgott. A comparison of solution strategies for biobjective shortest path problems. *Computers & Operations Research*, 36(4):1299–1331, April 2009.
10. R.E. Steuer and E. Choo. An interactive weighted Tchebycheff procedure for multiple objective programming. *Mathematical Programming*, 26(1):326–344, 1983.
11. C. Tung and K. Chew. A multicriteria Pareto-optimal path algorithm. *European Journal of Operational Research*, 62:203–209, 1992.
12. A.P. Wierzbicki. The use of reference objectives in multiobjective optimisation. In Fandel G. and Gal T., editors, *LNEMS 177*, pages 468–486. Springer Verlag, 1980.
13. A.P. Wierzbicki. On the completeness and constructiveness of parametric characterizations to vector optimization problems. *OR Spektrum*, 8:73–87, 1986.
14. M. Zeleny. *Multiple criteria decision making*, volume 123 of *LNEMS*. Springer, 1976.