GRACE: A Generational Randomized ACO for the Multi-objective Shortest Path Problem

Leonardo C. T. Bezerra[1⋆], Elizabeth F. G. Goldbarg[1⋆⋆], Luciana S. Buriol[2], and Marco C. Goldbarg[1⋆⋆⋆]

[1] Universidade Federal do Rio Grande do Norte, Natal, RN, Brazil
leo.tbezerra@gmail.com, beth@dimap.ufrn.br, gold@dimap.ufrn.br
[2] Universidade Federal do Rio Grande do Sul, Porto Alegre, RS, Brazil
buriol@inf.ufrgs.br

Abstract. The Multi-objective Shortest Path Problem (MSP) is a widely studied NP-Hard problem. A few exact algorithms were already proposed to solve this problem, however none is able to solve large instances with three or more objectives. Recently, some metaheuristics have been proposed for the MSP, but little can be said about their efficiency regarding each other, since no comparisons among them are presented in the literature. In this paper an Ant Colony Optimization (ACO) algorithm, called GRACE, is proposed for the MSP. The proposed approach is compared to the well-known evolutionary algorithm NSGA-II. Furthermore, GRACE is compared to another ACO algorithm proposed previously for the MSP. Results of a computational experiment with eighteen instances, with three objectives each, show that the proposed approach is able to produce high quality results for the tested instances.

Keywords: Shortest Path Problem, Multi-objective, Ant Colony Optimization.

1 Introduction

The Shortest Path Problem is a classical problem from graph theory and combinatorial optimization areas. Among several different real-world applications, routing scenarios are of particular interest. On the Internet, for example, determining the best route for sending a package following path-based protocols is an important step for ensuring routing efficiency. On navigation systems, which have become very common even on popular vehicles, shortest paths help on both planning and optimization of the resources. On emergency situations, retrieving minimal routes is critical to the rescue and survival of citizens.

Among the several shortest path problems (all pairs, point-to-point, k-shortest path, among others), in this study we refer to the Point-to-Point Shortest Path Problem simply as the Shortest Path Problem. Although much research has been

⋆ Leonardo C. T. Bezerra is supported by CNPq under project no. 579226/2008-5.
⋆⋆ Elizabeth F. G. Goldbarg is partially supported by CNPq.
⋆⋆⋆ Marco C. Goldbarg is partially supported by CNPq.

done on this problem [7, 4], usually real-world situations are more accurately represented via multiple criteria. Having more than one objective function to be optimized, multi-objective problems have a set of *non-dominated solutions*, instead of a single optimal solution. For a k-objective minimization problem (without loss of generality) we denote $s_1 \prec s_2$ (solution s_1 *dominates* solution s_2) iff $\forall i = 1, ..., k,\ s_1^i \leq s_2^i,$ and $\exists\, i,\ s_1^i < s_2^i$.

The non-dominated solutions are called *efficient solutions*, and the *Pareto optimal set* (or *Pareto set*) contains all such solutions. Each solution can be mapped to the *objective space* through its objective vector value. The set of objective vector values from the Pareto set is called *Pareto front*. Retrieving Pareto sets is the goal of multi-objective algorithms. Depending on the size of the instance, the nature of the problem, and the amount of objectives considered, this task might be unfeasible either by memory or time limitations. MSP, for instance, is an NP-Hard problem [30]. Thus, its Pareto set is expected to grow exponentially.

Several exact algorithms have been proposed for the MSP, and the most efficient relies on a two-phase strategy [28]. On the first phase, the algorithm searches for *supported efficient solutions*, which are solutions that can be retrieved by solving single objective problems obtained with weighted sums of the objectives. These solutions narrow the search space for the second phase, on which *non-supported efficient solutions* are searched (non-supported solutions cannot be retrieved via scalarizations). As far as the authors' knowledge concerns, no two-phase exact algorithms have been proposed for the MSP with more than two objectives.

Some metaheuristics have recently been proposed for solving the MSP using genetic algorithms and ant colony optimization. Their goal is to generate an *approximation set*, either containing suboptimal solutions, or part of the actual Pareto set, or even both. Some of the proposed approaches deal with the Bi-objective Shortest Path problem (BSP), but most focus on three or more objectives, and none uses two-phase strategies. Although several approaches have been proposed, solid multi-objective performance assessment has not been done to evaluate and compare them.

In this paper, a two-phase generational randomized Ant Colony Algorithm named *GRACE*, is proposed for the MSP. In order to find supported efficient solutions, a search strategy called *Logos* is proposed, which divides scalarization vectors in intervals, resembling divide-and-conquer techniques. The two- and three-objective versions of this strategy are detailed. GRACE is compared to two multi-objective algorithms: an ACO published by Häckel *et al.* for the MSP [14], and NSGA-II, a well-known multi-objective evolutionary algorithm proposed by Deb *et al.* [6] (an available implementation [25] was adapted for this problem).

The paper is organized as follows. Section 2 presents the MSP formulation and a literature review on algorithms for solving this problem. In Section 3, the Ant Colony Optimization approach is described, and several ACOs proposed for different multi-objective problems, including the MSP, are revised. GRACE and

Logos are presented in Section 4. The methodology used for the experimentation conducted in this work is described in Section 5, and results are discussed in Section 6. Finally, conclusions and future work are presented in Section 7.

2 The Multi-objective Shortest Path Problem

The Multi-objective Shortest Path problem studied in this paper is a generalization of the classical point-to-point shortest path problem, and is presented by Raith and Ehrgott using a network flow formulation for two objectives [28]. In this paper, we expand this formulation to deal with any number of objectives:

$$\min \quad z(x) = \begin{cases} z_1(x) = \sum\limits_{(i,j)\in A} c_{ij}^1 x_{ij} \\ \quad\quad ... \\ z_k(x) = \sum\limits_{(i,j)\in A} c_{ij}^k x_{ij} \end{cases} \tag{1}$$

$$\text{s.t.} \quad \sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = \begin{cases} 1 & \text{if } i = s, \\ 0 & \text{if } i \neq s, t \\ \text{-1} & \text{if } i = t, \end{cases} \tag{2}$$

$$x_{ij} \in \{0, 1\}, \quad \forall (i,j) \in A \ . \tag{3}$$

where s and t are, respectively, source and terminal nodes, c is a k-dimensional cost matrix for each edge (i, j), and z is the objective vector composed by k objective functions.

Several exact algorithms have been proposed for the MSP [11]. Skriver [32] proposed a classification for the particular case of two objectives (BSP), and grouped the algorithms as *node labelling* or *path/tree* algorithms. Among the node labelling algorithms, the existing methods were subdivided into two classes: *label setting* and *label correcting*. As for the path/tree family, the author identifies a *two-phase method* and a *k-th shortest path algorithm*. Recently, Raith and Ehrgott [28] compared the algorithms of the first three families using three sets of bi-criteria instances. Their experiment shows that label setting and correcting algorithms present good performance on smaller instances, whereas for large ones their times increase significantly. However, the dichotomic two-phase algorithm (different from the original two-phase algorithm [23]) is efficient even on large instances for the bi-objective case ($k = 2$).

Mooney and Winstansley [22] proposed a multi-objective evolutionary algorithm that has regular and elite populations, and generates the initial individuals through a random walking mechanism [5]. The authors use path encoding (representation by vertices), one-point crossover, binary tournament and a path mutation operator that substitutes the gene of a random locus by a random node. Both crossover and mutation operators only succeed in case the path feasibility is maintained. The experiments showed that the random walking strategy enabled the algorithm to cover a set of nodes and edges considered to be satisfactory by the authors. It also showed that, for three and four objectives, with instances

ranging from 100 to 3500 nodes, their algorithm was able to approximate a reference set created by three algorithms: Dijkstra, a k-th shortest path and many executions of their own algorithm. Finally, results show the EA was able to find extreme supported solutions faster than Dijkstra on real-world networks [3].

He *et al.* [13] also proposed an elitist evolutionary algorithm with two populations, but initial individuals are generated using depth first search mechanism and are ranked according to dominance ranking and niche count. They also use variable length chromosomes (path encoding using vertices), and an one-point crossover with binary tournament: a random cell is chosen from one parent, and in case the same node is also present on the other parent, the crossover is performed (a repair function eliminates possible loops). The mutation operator reconstructs the chromosome from a random locus through depth first search mechanism. An illustrative example showed the efficiency of the proposed MOEA on a 50 nodes three-objective instance.

Pangilinan and Janseens [27] tested SPEA2 [37] for the MSP using the implementation available on PISA framework [1]. They also used path encoding, but generated the initial population randomly. The one-point crossover and the mutation operator are identical to [13], but mutation reconstructs individuals randomly. Using three objectives instances, the approximation set presented good diversity on two of the objectives, but the authors were unable to determine the optimality of the solutions, since the actual Pareto sets were unavailable. Computational results showed that their MOEA is slower than [20].

Lin and Gen [18] presented a multi-objective evolutionary algorithm for the BSP. Their adaptive weight algorithm uses priority-based encoding (fixed chromosome sizes), roulette selection, weight-mapping crossover, a random mutation operator and also an immigration operator [21]. No information is given on how the initial population is generated. The authors also use two fuzzy logic controllers in order to auto-tune the parameters of their algorithm. They compare their MOEA against implementations of NSGA-II [6], SPEA [35] and rwGA [16] that use the same encoding and operators, analyzing diversity, cardinality, ratio of non-dominated solutions and computational time. Regarding these indicators, results show that the priority-based encoding and the auto-tuning are good strategies, and that their algorithm outperform the others.

As described above, several evolutionary algorithms have been proposed for the MSP over the last few years. However no comparison has been presented to evaluate whether one algorithm outperforms the others. In the next section, we revise the Ant Colony Optimization metaheuristic, and the literature on multi-objective ACOs, including the MSP.

3 Ant Colony Optimization

Ant Colony Optimization is a bio-inspired metaheuristic that uses the concept of *swarm intelligence* and *stigmergy*. It was originally proposed by Dorigo [10]

[3] These instances originally have only one objective. The author do not describe how and into how many objectives they were transformed.

as the *Ant System* (AS), and lately improved into *Ant Colony System* (ACS) [8]. The original AS consists of *agents* (ants) that are able to *evaluate their current state* and choose the next through a set of possible *transitions*. Each transition is weighted with *pheromone* and *heuristic* information (random proportional rule). The *constructive procedure* allows these agents to generate *viable solutions* for a combinatorial optimization problem. The ants are also able to *deposit* pheromone over the tracks they visit, which naturally *evaporates* with time. Two major changes were proposed to improve AS into ACS. First, when choosing the next state, a random value is tested against a threshold. Depending on that result, the choice might be for the best transition or using the original random proportional rule. Second, ants are allowed to update pheromone information as soon as they finish their constructive procedure. In AS, all ants perform pheromone update altogether when all of them finish building their solutions.

Dorigo and Socha [9] list some ACOs proposed for multi-objective problems. Iredi *et al.* [15] affirm some of these ACOs either consider some objectives to be more important than others, or deal with problems in which each objective can be treated separatedly. In this work, we limit ourselves to detailing the ACOs designed for problems with equally important objectives. In 2000, Iredi *et al.* [15] proposed a multi-colony ant system for a scheduling problem. In their work, multiple colonies are used to search different regions of the Pareto front. Each ant has a scalarization vector which is used for weighting heuristic information in the random proportional rule. Two pheromone matrices are used, one for each objective, and are updated as in ACS. Only ants that find new non-dominated solutions regarding a global archive are allowed to update the pheromone matrices. Lopéz-Ibanéz *et al.* [19] published a comparative study of different strategies for multi-objective ACO using local search strategies on the Bi-objective Quadratic Assignment Problem. The authors reported that results differ depending on the correlation of the objectives, but the usage of local search procedures improved the overall algorithm performance.

For the MSP, as far as the authors are aware of, only two ACOs have been proposed. Häckel *et al.* [14] proposed a version of the algorithm presented in [15] for the MSP. The algorithm proposed by Häckel et al. [14], referred to as HACO in this paper, do not allow two ants searching on the same region of the Pareto front (called overlapping zones [15]). The heuristic information comes from a Dynamic Programming algorithm called *Look-Ahead Heuristic* (LAH). As for the pheromone matrices, the authors present equations as if multiple matrices were used, but state that only one matrix is used. The experiments conducted in their work show that the usage of LAH improves overall results, and that their ACO obtains diverse solutions in comparison to a dynamic multi-objective algorithm not referenced by the authors on three objective instances[4].

Ghoseiri and Nadjari [12] proposed an ACO for the BSP and compared it to a label correcting algorithm, also not referenced. A single colony and two pheromone matrices (one for each objective) were used. As for the heuristic in-

[4] The authors do not explicit instances size, but state the edges plus vertices amount range from 527 to 1408.

formation, two sources were used: normalized edge weights and number of nodes to the destiny node. Scalarizations are used to compute both pheromone and heuristic information. This algorithm resembles ACS, except for the fact that local pheromone update is performed after each transition. The experimentation conducted by the authors used two objective NetMaker [31] instances. Results showed their algorithm was able to generate a diverse approximation set, but the Pareto fronts graphically presented no efficient solutions. Their ACO was computationally faster than the label correcting method. There is also no comparison between the proposed multiobjective ACOs for MSP and other metaheuristics.

4 GRACE: a two-phase ACO for the MSP

In this section, we present GRACE (Generational Randomized Ant Colony Enhancement) and the two-phase strategy (including the Logos procedures). The initial review on exact algorithms pointed that two-phase algorithms are efficient for the BSP, which led us to work on scenarios with three or more objectives. Since no two-phase strategy has been proposed for the shortest path problem with more than two objectives, we review the literature of other problems, and finally propose our own search strategy: a two-phase ACO for the MSP.

As seen in Section 3, many strategies have been proposed for multi-objectives ACO, but as reported by [19] the efficiency of the strategies depends much on the problem and on the instances. Hence, we chose a particular configuration and tested it against two algorithms from the literature. We chose one algorithm to represent each class of metaheuristics studied for the MSP. Since none of the MOEAs proposed had available implementations[5], we decided to use NSGA-II [6], a well-known MOEA with an available implementation for experimentation [25]. As for the ACOs, we implemented [14] according to the authors description.

4.1 Phase I: The Quest for Supported Efficient Solutions

In our review of search strategies for supported efficient solutions, we outline two methods found in the literature of MO problems with $k > 2$. Murata *et al.* [26] proposed a uniform distribution of scalarization vectors over the Pareto optimal surface. Their approach generates sequential weights separated by an ϵ gap. For large values of ϵ the algorithm is faster, but possibly miss many solutions, whereas for small values a larger set will be retrieved at the expense of higher computational times. Rocha [29] proposed a stochastic version of this algorithm, as not all of the weights are generated. Starting at the canonical scalarization vectors, the main objective scalarization value is decremented by ϵ and another random objective scalarization value is incremented by ϵ. The author found good results for the Multi-objective Quadratic Assignment Problem, but for large instances some parts of the Pareto front is not searched, resulting in loss of solutions.

[5] Each author was properly contacted through the emails listed on their papers.

In this work, we adapted the general idea present in the *geometric* [33] and the *dichotomic* [28] bi-criteria strategies in order to apply them to instances of three objectives. Both strategies start at the extreme supported efficient solutions s_1 and s_2, calculate an intermediary scalarization vector through evaluation functions, and test the existence of a new supported efficient solution s'. If the non-dominance of s' is confirmed, the search recursively proceeds for (s_1, s') and (s', s_2). Following the general idea, 2-Logos (Logarithmic 2-Objective Space Search) iteratively divides the region of the Pareto front, resembling a logarithmic function. Given two different non-dominated solutions s_i and s_f, found on (x_i, y_i) and (x_f, y_f) of the Pareto front respectively, the solution s_{mid}, from $(x_{midpoint}, y_{midpoint})$, is tested (lines 1-2). If it is a different non-dominated solution, the procedure is recursively called for (s_i, s_{mid}) and (s_{mid}, s_f) (lines 3-4). Otherwise, the recursion stops.

```
procedure 2-Logos
1:  {Require Pareto set S, solution si, solution sf};
2:  s_mid = Dijkstra(midpoint(si.weights, sf.weights));
3:  if (newSolution(S, s_mid)) then
4:      2-Logos(S, si, s_mid);
5:      2-Logos(S, s_mid, sf);
6:  end if
end
```

For three-objective instances, 3-Logos (Logarithmic 3-Objective Space Search) is proposed, and its pseudocode is presented below. 3-Logos initially finds extreme supported efficient solutions using Dijkstra algorithm. These three solutions are considered vertices of a triangle where each edge is formed by the line that connects each pair of these solutions (algorithm input). Each edge is scanned by the 2-Logos procedure (line 1). After completion of 2-Logos executions on the three edges (line 2), the centroid of the triangle is calculated and, in case the obtained solution is new in the set S of non dominated solutions (Pareto front), 3-Logos is recursively called for these three new subtriangles (lines 4-6). Both 2- and 3-Logos are illustrated in Figure 1.

```
procedure 3-Logos
1:  {Require Pareto set S, solution sa, solution sb, solution sc};
2:  S = addSolutions(S, 2-Logos(sa, sb), 2-Logos(sa, sc),
      2-Logos(sb, sc));
3:  s_ctr = Dijkstra(centroid(sa.weights, sb.weights, sc.weights));
4:  if (newSolution(S, s_ctr)) then
5:      3-Logos(S, sa, sb, s_ctr);
6:      3-Logos(S, sa, s_ctr, sc);
7:      3-Logos(S, s_ctr, sb, sc);
8:  end if
end
```

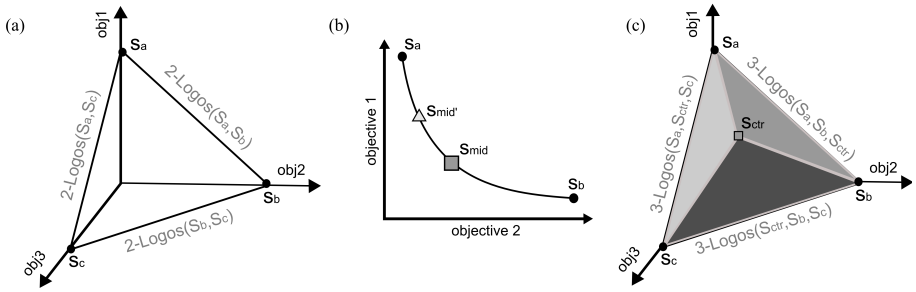

Fig. 1. Example of one iteration of 3-Logos. (a) The edges of the triangle are scanned through the 2-Logos procedure. (b) 2-Logos: if a new supported solution (s_{mid}) is found, recursively proceed for (s_a, s_{mid}) and (s_{mid}, s_b). (c) Once all edges have been scanned, 3-Logos recursively proceeds for (s_a, s_b, s_{ctr}), (s_a, s_{ctr}, s_c) and (s_{ctr}, s_b, s_c) if a new supported solution (s_{ctr}) exists.

4.2 Phase II: The Ant Colony

For the second phase of our ACO, we chose to create a single colony algorithm, in which iterations are called generations. Every ant is going to search the same region of the Pareto front using a scalarization vector randomly created for each generation. The purpose of this strategy is to allow a significant amount of ants to explore the same region. A single pheromone matrix is used in our ACO, and is warmed up at the end of Phase I. For every edge present in the efficient supported paths, a pheromone deposit $\tau_{deposit}$ is added, as many times as the edge appears in the set. An ant is only allowed to update the pheromone matrix when it finishes building its solution, and only if that solution is non-dominated regarding the non-limited global archive. For the heuristic information, we improved the idea used in [14] by using Dijkstra algorithm (we are assuming non-negative weight inputs). The heuristic information represents the distance, under a specific scalarization vector, from each node to the destination. The stopping criterion is a number R of generations with no generation of a new non-dominated solution regarding the global archive.

The pseudocode of GRACE is presented in the following page. Initially, extreme supported efficient solutions are found[6] (lines 1-2) and 3-Logos is called (line 3). The pheromone matrix is warmed up using the supported efficient solutions (line 4), and the generational cycle starts (line 5). First, a common scalarization vector is randomly created (line 6), and the heuristic information is calculated (line 7). Then, ants build their solutions (lines 8-14) and, if a new non-dominated solution regarding the global archive is found, a pheromone update is performed (lines 10-12). In case a new non-dominated solution was found over the last cycle, the algorithm is allowed to start a new one (line 15).

[6] In this work, we refer to the solutions found for scalarizations using canonical vectors as extreme supported efficient solutions.

```
Algorithm GRACE
1:  {Require graph G, vertex s, vertex t};
2:  extreme_solutions = Dijkstra(canonicalVectors());
3:  supported = addSolutions(supported, extreme_solutions);
4:  3-Logos(supported, extreme_solutions));
5:  pheromoneM = pheromoneWarmup(supported);
6:  for (i = 0; i < R; i++)
7:      l = randomScalarizationVector();
8:      heuristicM = heuristicInfo(Dijkstra(reverse(G), t, l));
9:      for (j = 0; j < n_ants; ++j)
10:         s = buildSolution(pheromoneM, heuristicM, s, t, l);
11:         if (newSolution(unsupported, s)) then
12:             unsupported = addSolution(unsupported, s);
13:             pheromoneUpdate(s);
14:         end if
15:     end for
16:     if ((i == R) and (newSolution(cycle))) then i = 0;
17: end for
end
```

5 Methodology

To compare GRACE against NSGA-II and the ACO of Häckel *et al.*[14], we adapted the existing NSGA-II implementation [25] for the MSP. We decided to use path encoding, which is also the choice made by most MOEAs proposed for this problem [13, 22, 27]. We also used crossover [13] and mutation [27] operators found in the MSP literature. The repair function of the crossover operator was redefined: the chromosome is scanned from the first to the last locus, and every time a duplicated vertex is found, all vertices between them are removed. We also used a specific data structure to store outgoing vertices and speed up the algorithm [2]. The algorithm of Häckel *et al.* [14], named HACO in this paper, was implemented following the directions given by the authors.

Since multi-objective optimizers produce approximation sets instead of single solutions, a specific methodology must be used for their evaluation. The guidelines for performance assessment utilized in this work are reviewed by Knowles *et al.* [17]. We use dominance ranking with the binary epsilon indicator [36] and, if necessary, the unary quality indicators I_H^1 [35] and $I_{\epsilon+}^1$ [36]. Reference sets are generated by merging all the approximation sets generated by the optimizers being compared, and removing the dominated solutions.

Tests were executed on an Intel Core 2 Duo @ 2.2GHz, with 1Gb of RAM and a Linux Ubuntu 9.04 distribution. We used a set of 18 instances, generated by Santos [24], from two distinct classes: *grid*, which represents a square grid, and *complete*, which contains complete graphs. For our experiment, we used fixed time limits per instance as the stopping criterium. In most cases, these limits were set according to instance size and class. Some details of the instances are

presented in Table 1, where column # shows the instance identification, *Type* shows the type of graph (complete, grid), $|N|$, $|A|$ and k are respectively the number of vertices, edges and objectives of each instance, and finally *t(s)* is the time limit in seconds for the executions of the algorithms.

Table 1. List of instances. Only the first three objectives were used on *large* instances.

#	*Type*	$\|N\|$	$\|A\|$	k	$t(s)$	*Seed*	#	*Type*	$\|N\|$	$\|A\|$	k	$t(s)$	*Seed*
1	CompleteN-small	25	600	3	5	45	10	GridN-small	64	224	3	7	14
2	CompleteN-small	50	2450	3	10	12	11	GridN-small	144	528	3	36	1
3	CompleteN-small	100	9900	3	12	13	12	GridN-small	256	960	3	81	26
4	CompleteN-medium	40	780	3	5	18	13	GridN-medium	484	1848	3	100	1
5	CompleteN-medium	120	14280	3	10	14	14	GridN-medium	961	3720	3	100	40
6	CompleteN-medium	200	39800	3	15	21	15	GridN-medium	1225	4760	3	100	2
7	CompleteN-large	100	9900	6	8	1	16	GridN-large	121	440	6	60	41
8	CompleteN-large	150	22350	6	40	10	17	GridN-large	484	1848	6	100	42
9	CompleteN-large	200	39800	6	40	1	18	GridN-large	900	3480	6	100	43

In order to fine-tune the NSGA-II, we tested several values for population size, number of generations, crossover and mutation rates. Since we used fixed time limits, we initially defined a large number of generations to find out the maximum population size the algorithm could stand: 200 generations for complete instances and 100 for grid instances. Complete instances are divided into two groups: instances 1 to 7 and instances 8 and 9, since results showed time limits were critical to population size. A linear function $pop(x) = a.t(x) + b$ was used to set the values of population size for grid instances. Table 2 shows the values used to each test configuration. Crossover rates 0.7, 0.8 and 0.9 were tested, as well as values 0.01, 0.005 and 0.001 for mutation rates [6]. Dominance ranking with the binary epsilon indicator [36] pointed population size of 2500 & 3000 to be statistically significantly better for complete instances. As for grid graphs, neither the dominance ranking nor the unary quality indicators I_H^1 [35] and $I_{\epsilon+}^1$ [36] were conclusive[7]. For crossover rates, no statistical significant differences was observed for complete instances, and for grid instances results were inconclusive. For mutation rates, no statistical significant differences was observed.

For HACO, we used the parameter setting presented by the authors in their paper. The only parameter not informed by the authors was the penalty parameter ξ. We tested several values ($\xi \in \{0.2, 0.4, 0.5, 0.6, 0.8\}$), but no statistical significant difference was found.

[7] When a configuration proved to be better for only few instances, we considered results to be inconclusive.

Table 2. Parameter values tested for NSGA-II.

#config	Complete 1-7	Complete 8-9	Grid a	Grid b
1	1000	1500	20	600
2	1500	2000	20	900
3	2500	3000	25	800

The final parameter settings were: (i) NSGA-II: *pop*=2500/3000, a=25, b=800, *cross*=0.9, *mut*=0.005; (ii) Hackel: $\chi_{colonies} = 3, \chi_{ants} = 4$, $\alpha = 0.5$, $\beta = 3$, $\tau_0 = 0.05$, $\rho = 0.1$, $q_0 = 0.5$, $\xi = 0.8$, and; (iii) GRACE: $n_{ants} = 300$, $\alpha = 0.6$, $\beta = 0.6$, $\tau_0 = 1$, $\tau_{deposit} = 10$, $R = 5$.

6 Results and Discussion

For comparing the results from the different algorithms, we perform a serie of dominance ranking [17] and statistical tests [3][34] over the results. As we can see in Table 3, the p-values of Kruskal-Wallis test on dominance ranking results indicate that there is a statistical significant difference between the dominance rankings of each optimizer. We then proceeded to the pairwise comparison using Wilcoxon test.

Table 3. p-values of Kruskal-Wallis test on dominance rankings from all optimizers.

#inst	p-value	#inst	p-value
1	**1.27884e-14**	10	**8.9668e-16**
2	**8.78175e-13**	11	**5.87126e-12**
3	**2.80243e-15**	12	**4.50099e-15**
4	**1.29004e-10**	13	**1.35497e-14**
5	**1.90311e-09**	14	**1.81209e-15**
6	**2.14250e-13**	15	**4.86612e-16**
7	**1.66174e-10**	16	**2.24666e-11**
8	**0.00508**	17	**1.22460e-14**
9	**3.09800e-13**	18	**3.97783e-16**

Table 4 shows p-values from both Wilcoxon test (the two-tailed and the one-tailed with "less" as hypothesis) for the comparison between GRACE and NSGA-II. p-values on the "less" column lower than 0.05 indicate that the dominance ranking of GRACE was better, whereas p-values greater than 0.95 indicate the dominance ranking of NSGA-II was better. For all instances (both grid and complete), results show GRACE generates statistically significantly better approximation sets than NSGA-II.

Table 4. p-values of Wilcoxon test on dominance rankings for GRACE and NSGA-II.

#inst	two-tailed	less	#inst	two-tailed	less
1	**1.09789e-09**	**5.49383e-10**	10	**8.79965e-11**	**4.39982e-11**
2	**3.64852e-10**	**1.82426e-10**	11	**6.28895e-08**	**3.14447e-08**
3	**1.86299e-10**	**9.31494e-11**	12	**0.01044**	**0.00522**
4	**3.98723e-09**	**1.99361e-09**	13	**0.00245**	**0.00012**
5	**1.65375e-07**	**8.26976e-08**	14	**4.20083e-09**	**2.100411e-09**
6	**3.08372e-10**	**1.54186e-10**	15	**4.87374e-11**	**2.43687e-11**
7	**1.31777e-08**	**6.58886e-09**	16	**4.27032e-11**	**2.13516e-11**
8	**0.02074**	**0.01037**	17	**0.00054**	**0.00027**
9	**1.50692e-10**	**7.53461e-11**	18	**9.31587e-11**	**4.365794e-11**

Next, we compare GRACE with HACO. Results on Table 5 show statistical significant difference for 13 instances, and in all of them the dominance rankings of GRACE are attested statistically better than the ones found by HACO. On the remaining 5 instances[8], unary quality indicators are necessary (Table 6). For the $I^1_{\epsilon+}$ indicator, GRACE approximation sets are considered better, but the I^1_H considers the opposite. This result means that the sets are incomparable [17].

Table 5. p-values of Wilcoxon test on dominance rankings for GRACE and HACO.

#inst	two-tailed	less	#inst	two-tailed	less
1	**1.03627e-09**	**5.18137e-10**	10	**3.31826e-11**	**1.65913e-11**
2	**0.03124**	**0.01562**	11	**1.13077e-09**	**5.65385e-10**
3	0.16151	0.92746	12	**2.77473e-12**	**1.38624e-12**
4	0.26094	0.13047	13	**2.77473e-12**	**1.38624e-12**
5	0.18886	0.90887	14	**4.43246e-12**	**2.21629e-12**
6	0.49546	0.24773	15	**1.34262e-11**	**6.71311e-12**
7	**0.01460**	**0.00730**	16	**3.42463e-08**	**1.71232e-08**
8	NaN	1	17	**2.77247e-12**	**1.38624e-12**
9	**1.51267e-05**	**7.56335e-06**	18	**4.43246e-12**	**2.21623e-12**

Results for the last comparison, between NSGA-II and HACO, are listed in Table 7. For complete instances, HACO generates better approximation sets for 8 out of 9 test-cases, whereas for grid instances NSGA-II generates better sets for all 9 instances.

[8] The dominance rankings from instance 8 were identical.

Table 6. p-values of Wilcoxon test on quality indicators for GRACE and HACO.

#inst	$I^1_{\epsilon+}$ two-tailed	$I^1_{\epsilon+}$ less	I^1_H two-tailed	I^1_H less
3	**5.34323e-08**	**2.67162e-08**	**1.58215e-14**	**1**
4	**2.71058e-10**	**1.35530e-10**	**1.32748e-09**	**1**
5	**9.79730e-11**	**4.89965e-11**	**1.21274e-09**	**1**
6	**6.59836e-10**	**3.29913e-10**	**1.58215e-14**	**1**
8	**7.63654e-08**	**3.81827e-08**	**1.58215e-14**	**1**

Table 7. p-values of Wilcoxon test on dominance rankings: NSGA-II [6], Hackel [14].

#inst	two-tailed	less	#inst	two-tailed	less
1	**0.03232**	**0.01616**	10	**5.72682e-10**	**2.86341e-10**
2	**4.02499e-07**	**0.99998**	11	**0.02284**	**0.01142**
3	**1.11944e-09**	**1**	12	**2.77247e-12**	**1.38624e-12**
4	**0.03177**	**0.98487**	13	**1.35016e-11**	**6.75080e-12**
5	**1.37130e-08**	**1**	14	**8.04777e-11**	**4.02388e-11**
6	**3.05784e-10**	**1**	15	**1.48317e-10**	**7.41586e-11**
7	**3.32396e-08**	**1**	16	**0.00116**	**0.00058**
8	**0.020744**	**0.99061**	17	**4.43246e-12**	**2.21623e-12**
9	**3.48046e-10**	**1**	18	**1.48645e-11**	**7.43227e-12**

7 Conclusions

In this work we reviewed the MSP literature, including exact algorithms, evolutionary algorithms, and ant colony optimization, and highlighted the lack of comparisons among the metaheuristics. We proposed a two-phase ACO, named GRACE, with a supported efficient solutions search called Logos.

For the evaluation of GRACE, we conducted a solid performance assessment against NSGA-II, a well-known MOEA and HACO, an ACO published for the MSP. For 18 instances from classes grid and complete, dominance ranking and statistical tests attested that GRACE generates better approximation sets than NSGA-II. In comparison with HACO, GRACE sets are better for 4 out of 9 complete instances, and 9 out of 9 grid. For the remaining complete instances, GRACE produces sets which are incomparable with HACO's according to two quality unary indicators, I^1_H and $I^1_{\epsilon+}$).

When comparing NSGA-II and HACO, the former generates statistically better sets than the latter for 1 out of 9 complete instances, and for all 9 grid instances. For the remaining complete instances, HACO was attested to generate better sets. As future work possibilities, a comparison including a MOEA proposed specifically for MSP could be used, as well as a larger instance database. Another promising work is the comparison of different multi-objective ACO strategies on the MSP, to test which configurations are more efficient.

References

1. Bleuler, S., Laumanns, M., Thiele, L., Zitzler, E.: Pisa — a platform and programming language independent interface for search algorithms. In: Fonseca, C., Fleming, P., Zitzler, E., Thiele, L., Deb, K. (eds.) EMO 2003, LNCS, vol. 2632, pp. 1–1. Springer Berlin / Heidelberg (2003)
2. Buriol, L.S., Resende, M.G.C., Thorup, M.: Speeding up dynamic shortest path algorithms. INF. J. on Comp. 20(2), 191–204 (2008)
3. Conover, W.J.: Practical non-parametric statistics. John Wiley and Sons, New York, NY, third edn. (1999)
4. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT Press and McGraw-Hill, second edn. (2001)
5. Costelloe, D., Mooney, P., Winstansley, A.: From random walks to pareto optimal paths. In: Proceedings of the 4th Irish Artificial Intelligence and Cognitive Science Conference. pp. 523–530 (2001)
6. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Trans. on Evol. Comp. 6(2), 182–197 (2002)
7. Dijkstra, E.W.: A note on two problems in connexion with graphs. Num. Math. (1959)
8. Dorigo, M., Gambardella, L.M.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans. on Evol. Comp. 1(1), 53–66 (1997)
9. Dorigo, M., Socha, K.: An introduction to ant colony optimization. Tech. Rep. TR/IRIDIA/2006-010, Institut de Recherches Interdisciplinaires et de Développements em Intelligence Artificialle, Université Libre de Bruxelles (2006)
10. Dorigo, M.: Optimization, learning and natural algorithms. Ph.D. thesis, Diparta-mento de Elettronica, Politecnico de Milano (1992)
11. Ehrgott, M., Gandibleux, X.: A survey and annotated bibliography of multiobjective combinatorial optimization. OR Spek. 22(4), 425–460 (2000)
12. Ghoseiri, K., Nadjari, B.: An ant colony optimization algorithm for the bi-objective shortest path problem. App. Soft Comp. 10(4), 1237–1246 (2010)
13. He, F., Qi, H., Fan, Q.: An evolutionary algorithm for the multi-objective shortest path problem. In: Proceedings of the Advances in Intelligent Systems Research (ISKE) (2007)
14. Häckel, S., Fischer, M., Zechel, D., Teich, T.: A multi-objective ant colony approach for pareto-optimization using dynamic programming. In: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO). pp. 33–40. ACM (2008)
15. Iredi, S., Merkle, D., Middendorf, M.: Bi-criterion optimization with multicolony ant algorithms. In: Zitzler, E., Thiele, L., Deb, K., Coello, C.A.C., Corne, D. (eds.) EMO 2001, LNCS, vol. 1993, pp. 359–372. Springer (2001)
16. Ishibuchi, H., Murata, T.: A multi-objective genetic local search algorithm and its application to flowshop scheduling. IEEE Trans. on Sys., Man and Cyber., Part C: App. and rev. 28(3), 392–403 (1998)
17. Knowles, J.D., Thiele, L., Zitzler, E.: A tutorial on the performance assessment of stochastic multiobjective optimizers. Tech. Rep. TIK-Report No. 214, Computer Engineering and Networks Laboratory, ETH Zurich (2006)
18. Lin, L., Gen, M.: A bicriteria shortest path routing problems by hybrid genetic algorithm in communication networks. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC). pp. 4577–4582 (2007)

19. López-Ibanez, M., Paquete, L., Stützle, T.: On the design of aco for the biobjective quadratic assignment problem. In: Dorigo, M., Birattari, M., Blum, C., Gambardella, L., Montada, F., Stützle, T. (eds.) ANTS 2004, LNCS, vol. 3172. Springer-Verlag (2004)
20. Martins, E.Q.V.: On a multicritera shortest path problem. Eur. J. of Op. Res. 16, 236 – 245 (1984)
21. Michael, C.M., Stewart, C.V., Kelly, R.B.: Reducing the search time of a steady state genetic algorithm using the immigration operator. In: Proceedings of the IEEE Congress on Tools for AI. pp. 500–501 (1991)
22. Mooney, P., Winstansley, A.: An evolutionary algorithm for multicriteria path optimization problems. Int. J. of Geo. Inf. Sci. 20(4), 401 – 423 (2006)
23. Mote, J., Murphy, I., Olson, D.L.: A parametric approach to solving bicriterion shortest path problems. Eur. J. of Op. Res. 53, 81 – 92 (1991)
24. MSPP: www.mat.uc.pt/∼zeluis/INVESTIG/MSPP/mspp.htm
25. Multi-objective NSGA-II code in C. Revision 1.1 (10 June 2005) (For Linux only): www.iitk.ac.in/kangal/codes.shtml
26. Murata, T., Ishibuchi, H., Gen, M.: Specification of genetic search directions in cellular multi-objective algorithms. In: Zitzler, E., Thiele, L., Deb, K., Coello, C.A.C., Corne, D. (eds.) EMO 2001, LNCS, vol. 1993, pp. 82–95. Springer (2001)
27. Pangilinan, J.M.A., Janseens, G.K.: Evolutionary algorithms for the multiobjective shortest path problem. Int. J. of Comp. and Inf. Sci. and Eng. 1 (2007)
28. Raith, A., Ehrgott, M.: A comparison of solution strategies for the biobjective shortest path problem. Comp. and Op. Res. 36(4), 1299–1331 (2009)
29. Rocha, D.A.M.: Busca local para o problema quadrático de alocação multiobjetivo (pqa-mo). Final essay for Computer Science B.Sc. Universidade Federal do Rio Grande do Norte, Natal, RN, Brazil (2006)
30. Serafini, P.: Some considerations about computational complexity for multi objective combinatorial problems. In: Jahn, J., Krabs, W. (eds.) Recent advances and historical development of vector optimization, Lecture Notes in Economics and Mathematical Systems, vol. 294. Springer Verlag, Berlin (1986)
31. Skriver, A.J.V., Andersen, K.A.: A label correcting approach for solving bicriterion shortest-path problems. Comp. and Op. Res. 27(6), 507 – 524 (2000)
32. Skriver, A.J.V.: A classification of bicriterion shortest path (bsp) algorithms. Asia-Pac. J. of Op. Res. 17, 199 – 212 (2000)
33. Steiner, S., Radzik, T.: Solving the biobjective minimum spanning tree problem using a k-best algorithm. Tech. Rep. TR-03-06, Department of Computer Science, King's College London (2003)
34. Wilcoxon, F.: Individual comparisons by ranking methods. Bio. bull. 1(6), 80–83 (1945)
35. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength pareto evolutionary algorithm. IEEE Trans. on Evol. Comp. 3(4), 257–271 (1999)
36. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., , da Fonseca, V.G.: Performance assessment of multiobjective optimizers: an analysis and review. Tech. Rep. TIK-Report No. 139, Institut für Technische Informatik und Kommunikationsnetze, ETH Zürich (2002)
37. Zitzler, E., Laumanns, M., Thiele, L.: Spea2: Improving the strength pareto evolutionary algorithm. Tech. Rep. TIK-Report 103, Computer Engineering and Networks Laboratory (TIK), Departament of Electrical Engeneering, Swiss Federal Institute fo Technology (ETH) Zurich (2001)