Expert Systems with Applications 40 (2013) 345–355

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Analyzing the impact of MOACO components: An algorithmic study on the multi-objective shortest path problem

Leonardo C.T. Bezerra [a], Elizabeth F.G. Goldbarg [a,*], Marco C. Goldbarg [a], Luciana S. Buriol [b]

[a] Universidade Federal do Rio Grande do Norte, Natal, RN, Brazil
[b] Universidade Federal do Rio Grande do Sul, Porto Alegre, RS, Brazil

ARTICLE INFO

Keywords:
Multi-objective shortest path
Ant Colony Optimization
Scalarization vectors

ABSTRACT

Multi-objective Ant Colony Optimization (MOACO) algorithms have been successfully applied to several multi-objective combinatorial optimization problems (MCOP) over the past decade. Recently, we proposed a MOACO algorithm named GRACE for the multi-objective shortest path (MSP) problem, confirming the efficiency of such metaheuristic for this MCOP. In this paper, we investigate several extensions of GRACE, proposing several single and multi-colony variants of the original algorithm. All variants are compared on the original set of instances used for proposing GRACE. The best-performing variants are also assessed using a new benchmark containing 300 larger instances with three different underlying graph structures. Experimental evaluation shows one of the variants to produce better results than the others, including the original GRACE, thus improving the state-of-the-art of MSP.

1. Introduction

Multi-objective Ant Colony Optimization (MOACO) algorithms have been studied for a decade now, and have been applied to several important multi-objective combinatorial optimization problems (MCOP) (Chica, Cordón, Damas, & Bautista, 2011; Doerner, Hartl, & Reimann, 2001; Doerner, Gutjahr, Hartl, Strauss, & Stummer, 2006; Iredi, Merkle, & Middendorf, 2001; Ke, Feng, Xu, Shang, & Wang, 2010; Mora, Merelo, Laredo, Millan, & Torrecillas, 2009). Given that a large amount of design possibilities can be devised, a number of experimental studies have been conducted aiming at analyzing individual algorithmic components and their contribution (García-Martínez, Cordón, & Herrera, 2007; López-Ibáñez, Paquete, & Stützle, 2004; López-Ibáñez & Stützle, 2010). The influence of multiple colonies, multiple heuristics and multiple pheromone structures are some of the most investigated topics.

In a previous work addressing the multi-objective shortest path problem (MSP), we have proposed a two-phase MOACO algorithm named GRACE (Bezerra, Goldbarg, Goldbarg, & Buriol, 2011) and showed its efficiency in a comparison to a well-known *evolutionary multi-objective* (EMO) algorithm, named NSGA-II (Deb, Pratap, Agarwal, & Meyarivan, 2002), and to another MOACO algorithm from the literature of MSP (Häckel, Fischer, Zechel, & Teich, 2008). Given the observed quality of the results, we continue such investigation by extending GRACE. Several single and multi-colony variants are proposed, using many of the efficient algorithmic components recently identified in the MOACO literature. To compare the variants, two sets of tri-criteria instances are used: (i) the original set of 18 instances with two different underlying graph structures used in the original GRACE (MSPP, 2010), ranging from 100 to 1000 nodes, and; (ii) a new instance benchmark proposed in this work containing 300 larger instances, ranging from 1000 to 8000 nodes, with three different underlying graph structures. All comparisons use solid assessment methodology: dominance rankings (Knowles, Thiele, & Zitzler, 2006), unary Pareto-compliant quality indicators (Zitzler, Thiele, Laumanns, Fonseca, & da Fonseca, 2003), and non-parametrical statistical tests (Conover, 1999; Holland, 1975). Results show the performance of the variants highly depends on the underlying graph structure for smaller instances. For larger instances, one of the variants outperforms the others as well as the original GRACE algorithm, thus establishing a new state-of-the-art for the problem.

The contribution of this work, however, is not limited to an experimental study of existing algorithmic components or to the improvement of the state-of-the-art of an important MCOP. Over the literature, one of the key aspects that has not yet been investigated is how to generate and assign scalarization vectors to colonies in a MOACO algorithm. Scalarization vectors play a central role in multi-objective optimization, allowing algorithms to deal with single objective versions of the problem, which are generally less difficult to solve. Efficient multi-objective algorithms make use of such approach, regardless of the metaheuristic employed (Paquete & Stützle, 2003; Vianna & Arroyo, 2004; Zhang & Li,

* Corresponding author. Tel./fax: +55 8436421218.
E-mail addresses: leo@ppgsc.ufrn.br (L.C.T. Bezerra), beth@dimap.ufrn.br (E.F.G. Goldbarg), gold@dimap.ufrn.br (M.C. Goldbarg), buriol@inf.ufrgs.br (L.S. Buriol).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2012.07.052

2007). Traditionally, scalarization vectors have only been generated *randomly* in the [0,1] space, or *systematically*, parameterized by a number of divisions. In this work, all the variants proposed use a different method to generate or assign such vectors, many of them novel. In fact, the variant that outperforms the others differs from the original GRACE only on the way scalarization vectors are generated and assigned to colonies.

This paper is organized as follows. Section 2 describes the MSP and reviews its state-of-the-art. In Section 3, the ACO metaheuristic is revised and the different approaches found over the literature are detailed. In Section 4, GRACE and its single-species variants are described and compared. In Section 5, the experimental setup for the comparison of the variants on the original set of instances is described, and results are presented and discussed. In Section 6, the new instance benchmark is proposed, listing results from an exact algorithm. The comparison between the two best performing variants are also presented in this section. Finally, conclusions and future work possibilities are discussed 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 (Raith & Ehrgott, 2009). The expanded version of this formulation to deal with any number of objectives is used in this work:

$$\min\ z(x) = \begin{cases} z_1(x) = \sum_{(i,j)\in A} c^1_{ij} x_{ij}, \\ \cdots \\ z_k(x) = \sum_{(i,j)\in A} c^k_{ij} x_{ij}, \end{cases} \quad (1)$$

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

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

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.

The MSP belongs to a class of problems that have polynomial algorithms for their original single objective versions. This allows optimizers to use a two-phase strategy that was shown to perform efficiently on the bi-objective context (Paquete & Stützle, 2003; Raith & Ehrgott, 2009). This approach takes advantage of the fact that some of the solutions contained in the Pareto set can be retrieved with the help of *scalarizations*, i.e., by attributing weights to each objective. This means that part of the Pareto set can be retrieved polynomially. Having these solutions (or part of them) *a priori*, an optimizer can narrow the search space when looking for non-supported solutions. The literature on this problem includes exact and heuristic algorithms, which include evolutionary and ant colony optimizers. Raith and Ehrgott, 2009 recently compared different exact proposals (Martins, 1984; Mote, Murphy, & Olson, 1991; Skriver & Andersen, 2000) for the Bi-objective Shortest Path (BSP) problem, and showed that the two-phase approach is able to solve instances up to 21000 nodes in less than 30 s, and up to 300000 nodes in less than 40 s. This algorithm, however, has not been expanded to handle problems with more than two objectives (this generalization is not trivial).

Given the excellent results of exact algorithms for the BSP, our review of the MSP literature does not include algorithms proposed for dealing with two objectives. Among the EMO algorithms proposed for the MSP (He, Qi, & Fan, 2007; Mooney & Winstansley, 2006; Pangilinan & Janseens, 2007), some commonalities can be identified. First, the elitist approach can be identified in all, with each algorithm keeping two populations during its entire execution. Second, selection by binary tournament is used by all algorithms. Finally, crossover operators differ on the method to exchange genes, but all use a one-point approach. Important different features are listed in Table 1. We refer to the traditional exchange of genes used in evolutionary algorithms crossover operators as *direct exchange*. The paired exchange used by He et al. (2007) and Pangilinan and Janseens (2007) is a four-stage procedure: (i) a locus i is randomly chosen in one of the parents (p_1); (ii) the operator searches the other parent (p_2) to find out if the same gene is present; (iii) if the gene is found in locus j of p_2, a direct exchange is performed considering positions i and j as starting points for p_1 and p_2 respectively, and finally; (iv) if a cycle is present in any of the resulting offsprings, a repair function is called to remove it by discarding all elements of the cycle. Concerning mutation operators, the algorithm proposed by Mooney and Winstansley (2006) randomly chooses one locus of the chromosome, and replaces its gene with a randomly chosen node. This operation is only considered to succeed if a feasible path is generated. By their turn, He et al. (2007) and Pangilinan and Janseens (2007) reconstruct part of the chromosome, starting at a randomly choose locus. The methods applied for the reconstruction of the chromosomes can be seen in Table 1. Since no common experimental setup is used by the authors, the actual performance of the algorithms remains unclear.

To the best of our knowledge, only one MOACO algorithm (other than GRACE) has been proposed for the MSP (Häckel et al., 2008), and will be detailed in the next section.

3. Ant Colony Optimization

Ant Colony Optimization (ACO) is a bio-inspired metaheuristic that uses the concept of *swarm intelligence*, i.e., the ability of groups to communicate even in the absence of a central coordination, through a stimulus that is at the same time physical and local. This phenomenon is called *stigmergy* (Grass é, 1959), and in the case of ants, this stimulus is called *pheromone* (Dorigo & Socha, 2006). When looking for food, ants are able to identify and perform pheromone deposits over the paths they tread in order to guide other ants to follow their trails. Since pheromone evaporates with time, shorter paths are more likely to be reinforced than longer ones, promoting convergence towards promising solutions.

In an attempt to mimic this natural behavior, the *Ant System* (AS) metaheuristic was proposed (Dorigo, 1992), later improved into the *Ant Colony System* (ACS) (Dorigo & Gambardella, 1997) and the $\mathcal{MAX}-\mathcal{MIN}$ Ant System (MMAS) (Stützle & Hoos, 2000). In all algorithms, solutions are iteratively built by agents called *ants*. Each ant constructs a solution and is able to evaluate its current state, i.e., to calculate the objective value of its current path. While building its solution, an ant makes decisions at each state based on the information available for the possible choices. Such information comes from the experience of other ants, by means of the amount of pheromone deposited on a given path, and from its own experience, by means of a heuristic. The probability that an ant makes the transition from state i to state j, $p(e_{ij})$, is given in Eq. (4) where $\mathcal{N}_i$ denotes the set of states that are reachable from i, π_{ij} denotes the amount of pheromone between states i and j, η_{ij} denotes the heuristic value associated to the transition to state j, and α and β are parameters that weight the importance of the information that comes from the pheromone and the heuristic, respectively.

Table 1
Components of EMO algorithms proposed for the MSP.

Algorithm	Initial population	Crossover	Mutation
Mooney and Winstansley (2006)	Random walking	Direct exchange	–
He et al. (2007)	Depth-first search	Paired exchange	Depth-first search
Pangilinan and Janseens (2007)	Random	Paired exchange	Random

$$p(e_{ij}) = \begin{cases} \frac{\tau_{ij}^{\alpha}\cdot\eta_{ij}^{\beta}}{\sum_{h\in\mathcal{N}_i}\tau_{ih}^{\alpha}\cdot\eta_{ih}^{\beta}} & \text{if } j \in \mathcal{N}_i, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

At the end of each iteration, pheromone *deposits* may be performed, which naturally *evaporates* over time. The major differences between AS, ACS and MMAS refer to pheromone limits and update: (i) in ACS, a local pheromone deposit may be performed, as soon as an ant finishes constructing its solution, and; (ii) in MMAS, the pheromone information is subject to limits, τ_{max} and τ_{min}, and initially set to τ_{max}. Moreover, both MMAS and ACS use a pseudo-random proportional rule, i.e., ants may choose the best transition available instead of using stochasticity. In this paper, all algorithms implemented are based on the ACS, but the original transition rule from AS is adopted.

3.1. Multi-objective ACOs

Several proposals can be found on the literature of ACO algorithms for multi-objective problems. These MOACO algorithms show significant differences regarding important design questions. Since a large number of proposals have been made, we review them according to the following three topics:

3.1.1. Solution construction

Since the nature of the objectives for a given problem may be heterogeneous, some algorithms choose to use objective-specific heuristics (Alaya, Solnon, & Ghédira, 2007; Barán & Schaerer, 2003; Doerner et al., 2001, 2006; Doerner, Gutjahr, Hartl, Strauss, & Stummer, 2004; Ghoseiri & Nadjari, 2009; Häckel et al., 2008; Mora et al., 2009) and/or pheromone structures (Alaya et al., 2007; Barán & Schaerer, 2003; Doerner et al., 2001; Ghoseiri & Nadjari, 2009; Häckel et al., 2008; Iredi et al., 2001; Ke et al., 2010; Mora et al., 2009). These different sources of information need to be aggregated, traditionally by means of scalarization vectors. The commonly employed aggregation methods over the literature include *weighted sum* (Doerner et al., 2001, 2004, 2006; Mora et al., 2009), *weighted product* (Alaya et al., 2007; Ghoseiri & Nadjari, 2009; Iredi et al., 2001) and *non-weighted sum* (Alaya et al., 2007). Another possible aggregation approach is to *randomly* choose which objective to optimize at each iteration or step of the constructive procedure (Alaya et al., 2007). Furthermore, algorithms differ as to use a single scalarization vector per iteration (López-Ibáñez & Stützle, 2010), or many scalarization vectors simultaneously (Chica et al., 2011; Doerner et al., 2004, 2006; Iredi et al., 2001).

3.1.2. Number of colonies

The most straightforward generalizations of ACOs to deal with multi-objective problems maintain traditional single colony approaches (Alaya et al., 2007: Barán & Schaerer, 2003; Doerner et al., 2004, 2006; Ghoseiri & Nadjari, 2009; Ke et al., 2010; Mora et al., 2009). Others, more elaborated, choose to use more than one colony in an attempt to specialize in different regions of the Pareto front. Theoretically, such specialization requires: (i) having (at least) one pheromone structure per colony; (ii) using only scalarization vectors that represent the region for which the colony is responsible, and; (iii) reinforcing the pheromone structures only with solutions that belong to the region for which the colony is responsible. Over the literature, condition (i) is not always used, since some algorithms use different colonies with common pheromone structures (Alaya et al., 2007; Häckel et al., 2008; Ke et al., 2010). Concerning the scalarization vectors used by each colony, colonies may *share* scalarization vectors (Iredi et al., 2001) or use complete *disjoint* sets of vectors (Chica et al., 2011; Iredi et al., 2001). Finally, concerning the regionalized selection for pheromone update, this approach has only been proposed so far for bi-criteria problems (Iredi et al., 2001). The other method used over the literature is the traditional update by origin, that is, if solution *i* was found by colony *j*, it will be used to update the pheromone structure (s) of *j*.

3.1.3. Selection criteria for pheromone update

The two major selection criteria for pheromone update currently found on the literature allow either the *best-per-objective* or the *nondominated* solutions to deposit pheromone. In the first case, a number of best-per-objective ants (Alaya et al., 2007; Doerner et al., 2001, 2004, 2006) is allowed to update. In the second, only ants that found nondominated solutions (Alaya et al., 2007; Barán & Schaerer, 2003; Ghoseiri & Nadjari, 2009; Häckel et al., 2008; Iredi et al., 2001; Ke et al., 2010; Mora et al., 2009) can perform deposits. In both cases, the reference used for the selection of best-per-objective or nondominated solutions vary according to: (i) *colonies*, which means solutions can be compared with a global (Ghoseiri & Nadjari, 2009; Iredi et al., 2001; Ke et al., 2010; Mora et al., 2009) or a local (Alaya et al., 2007; Doerner et al., 2004, 2006; Häckel et al., 2008; Iredi et al., 2001) pool of solutions, and; (ii) *history*, that is, solutions can be compared only with the solutions found in a given iteration (Doerner et al., 2004, 2006; Ghoseiri & Nadjari, 2009; Iredi et al., 2001) or to a history of best-so-far solutions (Häckel et al., 2008; Iredi et al., 2001; Ke et al., 2010; Mora et al., 2009).

3.2. MOACOs for the MSP

To the best of our knowledge, only two MOACO algorithms have been proposed so far for the MSP (Ghoseiri & Nadjari, 2009; Häckel et al., 2008). However, since the MOACO proposed by Ghoseiri and Nadjari (2009) addresses the particular case of $k = 2$ (BSP), we will limit ourselves to the description of the algorithm proposed by Häckel et al. (2008). The MOACO algorithm proposed by Häckel et al. (2008) uses a multi-colony approach, with disjoint intervals of weights. A single global pheromone matrix is adopted. Selection by dominance is used, as well as the update by origin method. The heuristic information comes from a Dynamic Programming algorithm called *Look-Ahead Heuristic* (LAH), proposed by the authors. The experiments conducted in their work show that using LAH improves overall results, and that their ACO obtains solutions well distributed along the tri-dimensional objective space in comparison to a dynamic multi-objective algorithm not referenced by the authors. However, in a previous work we have compared our implementation of this algorithm with GRACE and the latter outperformed the first under the experimental setup reproduced here.

4. Extending GRACE

Many are the possibilities when designing a MOACO algorithm. In this section, several extensions are proposed and tested for

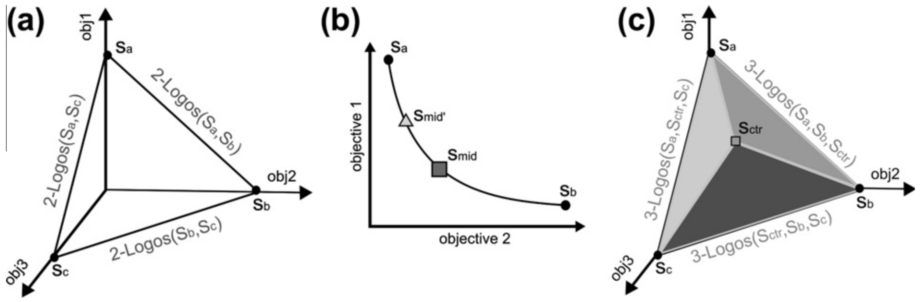

Fig. 1. Example of one iteration of 3-Logos.

GRACE (Bezerra et al., 2011). The original algorithm is described in Section 4.1. The proposed variants are presented in Section 4.2.

4.1. GRACE

Devised as a two-phase algorithm, GRACE initially uses a search procedure, named *Logos*, to retrieve supported efficient solutions. This search strategy can be used for bi and tri-criteria scenarios (2 and 3-Logos, respectively). 2-Logos (see Algorithm 2-Logos below) iteratively divides each search region into 2 sub-regions, resembling a logarithmic function. The general framework of 2-Logos is presented below. In the first execution of 2-Logos two solutions s_1 and s_2 are obtained by solving the single-objective minimum shortest path for each objective separately. Each solution corresponds to solving an scalarized version of the MSP problem with weights (1,0) and (0,1). More generally, solutions s_1 and s_2 correspond respectively to points (x_1,y_1) and (x_2,y_2) in the objective space. A scalarization vector corresponding to $(x_{midpoint}, y_{midpoint})$ is generated, and the single-objective minimum shortest path using this ponderation is retrieved, returning a solution s_{mid}. If this solution is non-dominated regarding the global archive, the procedure is recursively called for (s_1,s_{mid}) and (s_{mid},s_2). This strategy is similar to dichotomic approaches (Raith & Ehrgott, 2009; Steiner & Radzik, 2003).

Analogously, 3-Logos (see Algorithm 3-Logos below) initially finds the extreme supported efficient solutions using Dijkstra's algorithm (Dijkstra, 1959), that is, the solutions for the MSP considering each objective separately. These three solutions are considered vertices of a triangle where each edge is formed by the line that connects each pair of input weights. The edges are initially scanned by 2-Logos. Then, a scalarization vector λ_{ctr} corresponding to the centroid of the triangle comprised by the input weights is calculated, in a similar fashion to the procedure adopted for 2-Logos. At this stage, Dijkstra's algorithm is run on the scalarized MSP considering λ_{ctr}. In case the obtained solution is new in the set S of non dominated solutions, 3-Logos is recursively called for three new sub-triangles, as shown in Fig. 1. The pseudocode of 3-Logos can be seen below.

Algorithm: *2*-Logos (extreme solutions s_1 and s_2)

```
generate new scalarization vector: $\lambda^i_{mid} = \frac{\lambda^i(s_1)+\lambda^i(s_2)}{2}$
find $s_{mid}$ using Dijkstra and $\lambda_{mid}$
if $s_{mid} \neq s_1$ and $s_{mid} \neq s_2$, and $s_{mid}$ is non-dominated w.r.t
  global archive
  add $s_{mid}$ to the global archive
  call 2-Logos for $(s_1,s_{mid})$
  call 2-Logos for $(s_{mid},s_2)$
```

Algorithm: *3*-Logos (extreme solutions s_a,s_b,s_c)

```
call 2-Logos for $(s_1,s_2)$
call 2-Logos for $(s_1,s_3)$
call 2-Logos for $(s_2,s_3)$
generate new scalarization vector: $\lambda^i_{ctr} = \frac{\lambda^i(s_a)+\lambda^i(s_b)+\lambda^i(s_c)}{3}$
find $s_{ctr}$ using Dijkstra and $\lambda_{ctr}$
if $s_{ctr} \neq s_a \neq s_b \neq s_c$, and $s_{ctr}$ is non-dominated w.r.t global
  archive
  add $s_{ctr}$ to the global archive
  call 3-Logos for $(s_1,s_2,s_{ctr})$
  call 3-Logos for $(s_1,s_{ctr},s_3)$
  call 3-Logos for $(s_{ctr},s_2,s_3)$
```

The pseudocode of the second phase of GRACE can be seen below. A single colony approach is adopted. The solutions found in the first phase are used to warm up the single pheromone matrix: every time an edge is found in a supported efficient solution, a deposit is made. Dijkstra's algorithm provides heuristic information to the ants: a scalarization vector is used to obtain a single-objective problem, and Dijkstra's algorithm is applied on an inverted graph, which is obtained from the original graph with the inversion of the direction of all edges. In other words, the heuristic information available in each node v is the distance from v to the terminal node t, under a given scalarization. These scalarization vectors are randomly generated at the beginning of each iteration, and all ants use the same vector per iteration. New non-dominated solutions found by the ants are added to an external archive, which is unlimited. The ants are only allowed to perform pheromone updates if they find new non-dominated solutions. Each of the ants allowed to update the pheromone matrix deposits an equal amount of pheromone, regardless of the objective values of the solution found. The main loop comprises a group of R iterations each called *cycle*. If during a cycle a new non-dominated solution arises, then a new cycle of iterations is allowed and the algorithm receives more processing time.

Algorithm: GRACE (graph G, nodes s and t)

```
find initial solutions using Logos and Dijkstra
warmup pheromone structure using initial solutions
repeat
  generate a random scalarization vector
  calculate heuristic information using Dijkstra
  for each ant $a$ in 1..#Ants
    construct a solution $sol$
    if $sol$ is new and non-dominated
      add $sol$ to the global archive
      update pheromone
until a cycle cannot find a new non-dominated solution
```

4.2. Proposed extensions

Extensions of this algorithm are presented in the following subsections, with single and multiple-colony approaches. When the concept behind the version does not allow the single colony approach, it is explicitly stated. In the multi-colony approaches, different colonies do not interact directly: each of them has its own pheromone matrix and heuristic information. A single type of heuristic information and pheromone is used for all colonies. However, when selecting ants to perform pheromone update, all colonies test their solutions according to dominance in regard to the external global archive that stores the solutions found by all colonies during the whole execution of the algorithm. Regarding the update by region strategy, an important question arises: how to order solutions to determine regions when this update strategy is used on problems with three objectives. Whereas a simple lexicographic ordering divides the objective space *"equally"* among objectives in the bi-objective case, there is no trivial solution for the three objective context. The strategies used to overcome this problem are described later.

Five variants of the basic algorithm are proposed differing from each other basically on scalarization vector used to provide heuristic information to the ants. These versions are named: *Global-Random*, *Pseudo-Global-Random*, *Fixed-Unique*, *Fixed-Multi*, and *Objective-Driven*. Moreover, two hybrid versions named *Mixed* and *Mixed Objective-Driven* are also proposed.

4.2.1. Global-Random

The first variant, called *GRan*, uses randomly generated scalarization vectors, as the original GRACE algorithm. However, multiple colonies may be used, in which case the *update by origin* method is adopted. The pseudocode of GRan can be seen below:

Algorithm: Variant GRan

```
for each colony c in 1..#Colonies
  randomly generate a scalarization vector $\lambda$
  for each ant a in 1..#Ants
    construct a solution sol using $\lambda$
    if sol is a new non-dominated solution
      update pheromone structure of colony c
```

4.2.2. Pseudo-Global-Random

The second variant, called PGRan, generates random scalarization vector within triangular bounds. Initially, all possible scalarization vectors are generated, given an arbitrary number d of divisions. For example, for $k = 3$ and $d = 2$, six vectors are created: [1,0,0], [0.5,0.5,0], [0.5,0,0.5], [0,0.5,0.5], [0,1,0] and [0,0,1]. These vectors comprise triangles on the objective space, as depicted in Fig. 2. In this variant, these triangular regions are considered colonies, and at each iteration a random scalarization vector bounded by the colonies' triangles are generated. The pseudocode of PGRan can be seen below. When multiple colonies are used, the update by origin method is adopted.

Algorithm: Variant PGRan

```
generate set of vectors systematically
for each colony c in 1..#Colonies
  generate random vector $\lambda$ bounded by colony limits
  for each ant a in 1..#Ants
    construct a solution sol using $\lambda$
    if sol is a new non-dominated solution
      update pheromone structure of colony c
```

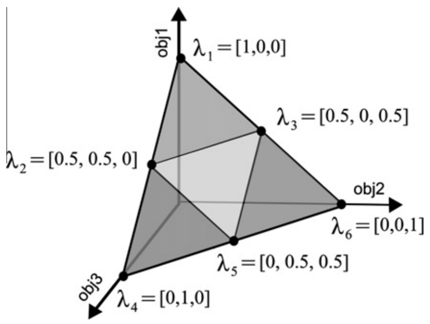
obj1
$\lambda_1 = [1,0,0]$
$\lambda_3 = [0.5, 0, 0.5]$
$\lambda_2 = [0.5, 0.5, 0]$
obj2
$\lambda_6 = [0,0,1]$
$\lambda_5 = [0, 0.5, 0.5]$
obj3
$\lambda_4 = [0,1,0]$

Fig. 2. Colonies in PGRan ($k = 3, d = 2$).

4.2.3. Fixed-Unique

The third variant, called FUn, generates the scalarization vectors systematically given an arbitrary number d of divisions, as in the generation of bounds for PGRan. In this variant, each colony is assigned a single scalarization vector, and the update by origin method is used. This variant cannot be implemented with only one colony, because when $d = 1$ the number of colonies equals the number of objectives, which is always greater than one. The pseudocode of FUn can be seen below, where λ_c is the cth scalarization vector.

Algorithm: Variant FUn

```
generate set of vectors systematically
for each colony c in 1..#Colonies
  for each ant a in 1..#Ants
    construct solution sol using $\lambda_c$
    if sol is a new non-dominated solution
      update pheromone structure of colony c
```

4.2.4. Fixed-Multiple

The fourth variant, called FMu, also generates the scalarization vectors systematically given an arbitrary number d of divisions, as in FUn. In this variant, to each colony multiple scalarization vectors are assigned. The distribution of vectors is performed in the order they were generated. Hence, for $m = 3$, $k = 3$, and $d = 2$, colony 1 is assigned vectors [1,0,0] and [0.5,0.5,0], colony 2 is assigned vectors [0.5,0,0.5] and [0,0.5,0.5], and colony 3 is assigned vectors [0,1,0] and [0,0,1]. If the number of vectors is not divisible by the number of colonies, the last colony receives the extra vectors. This variant can be implemented using single or multiple colonies. When multiple colonies are used, the update by origin method is adopted. The pseudocode of FMu can be seen below, where λ_c^v is the vth vector assigned to colony c.

Algorithm: Variant FMu

```
generate set of vectors systematically
sequentially assign equal amount of vectors to colonies
for each colony c in 1..#Colonies
  for each vector v in 1..#Vectors_per_colony
    for each ant a in 1..#Ants
      construct solution sol using $\lambda_c^v$
      if sol is a new non-dominated solution
        update pheromone structure of colony c
```

4.2.5. Mixed

The fifth variant, named Mix, is identical to FMu, except for the inclusion of a random scalarization vector in each colony. Single or multiple colonies may be used. When multiple colonies are adopted, the update by origin is used. The pseudocode of Mix can be seen below, where λ_c^v is the vth vector assigned to colony c.

Algorithm: Variant Mix

```
generate set of vectors systematically
sequentially assign equal amount of vectors to colonies
for each colony $c$ in 1..#Colonies
  for each vector $v$ in 1..#Vectors_per_colony
    for each ant $a$ in 1..#Ants
      construct solution sol using $\lambda_c^v$
      if sol is a new non-dominated solution
        update pheromone structure of colony $c$
  randomly generate a scalarization vector $\lambda_{random}$
  for each ant $a$ in 1..#Ants
    construct a solution sol using $\lambda_{random}$
    if sol is a new non-dominated solution
      update pheromone structure of colony $c$
```

4.2.6. Objective-Driven

The sixth variant, named Obj, has colonies specialized on each objective. The scalarization vectors are generated as in FUn. For an instance with k objectives, a given colony i, $i = 1,\ldots,$#Colonies, is assigned all vectors that priorize criterion $1 + (i-1)\%k$ (one plus the remainder of the integer division of $i-1$ by k). For example, a vector [0.5,0.5,0] would be rejected by colony 1, for the weight associated with the first objective is not greater than 0.5. Since there must be at least one colony per objective, a single colony version cannot be created. However, there is no limitation in how many colonies there might be per objective. Throughout this paper, only one colony per objective is used. Each colony has its own pheromone matrix, that specializes for its primary objective.

For the pheromone update, three strategies were tested: *update by origin*, *update by region* and using both strategies at the same time. In the case of the update by region strategy it is necessary to determine the region to which a new generated solution belongs. Given a new solution s_{new} and an external archive A, the policy adopted states that: (i) if a solution $s \in A$ is dominated by the new solution s_{new}, then the pheromone matrices of the colonies that focus on the objectives that have been optimized are updated, or; (ii) if s_{new} does not dominate any solution of A, it is compared with the last solution found. For every objective $i \in \{1,\ldots,k\}$, if $f^i(s_{new}) < f^i(s_{last})$, a pheromone update is performed on the matrix of the colony that focuses that objective.

When update by origin and update by region are used simultaneously, the strategy applied to determine which regions a solution belongs to is used first. The region corresponding to the colony the ant that generated that solution belongs to is always updated. The pseudocode of Obj is shown below.

Algorithm: Variant Obj

```
generate set of vectors systematically
sequentially assign equal amount of vectors to colonies
for each colony $c$ in 1...#Colonies
  for each vector $v$ in 1...#Vectors
    if component $(1 + (c-1)\%k)$ of $\lambda_v > 0.5$
      for each ant $a$ in 1...#Ants
        construct solution sol using $\lambda_v$
        if sol is a new non-dominated solution
          determine regions of sol
          if update by origin then update colony $c$
          if update by region then update regions
```

4.3. Mixed Objective-Driven

The seventh variant, called OMix, is basically Obj with the inclusion of a random scalarization vector (similar to Mix and FMu). The pseudocode for OMix is shown below.

Algorithm: Variant Obj

```
generate set of vectors systematically
sequentially assign equal amount of vectors to colonies
for each colony $c$ in 1...#Colonies
  for each vector $v$ in 1...#Vectors
    if component $(1 + (c-1)\%k)$ of $\lambda_v > 0.5$
      for each ant $a$ in 1...#Ants
        construct solution sol using $\lambda_v$
        if sol is a new non-dominated solution
          determine regions of sol
          if update by origin then update colony $c$
          if update by region then update regions
  randomly generate a scalarization vector $\lambda_{random}$
  for each ant $a$ in 1...#Ants
    construct a solution sol using $\lambda_{random}$
    if sol is a new non-dominated solution
      determine regions of sol
      if update by origin then update colony $c$
      if update by region then update regions
```

5. Experiments with the original instance set

In this section, we perform a comparison among all the proposed variants using the original instance set used by Bezerra et al. (2011). First, Section 5.1 presents the experimental setup used, namely the set of instances, performance assessment methodology and platform description. The parameter calibration performed for each variant is described in Section 5.2. Finally, results are presented and discussed in Section 5.3.

5.1. Methodology, platform and instances

The assessment of the results produced in the computational experiments reported in this paper is done based on the methodology presented by Knowles et al. (2006). At first, the approximation sets delivered by the tested algorithms are compared by means of dominance rankings. Given a list of q optimizers, r_i $(i = 1,\ldots,q)$ independent executions for each of them and a collection $\mathbf{C}$ containing all approximation sets C_j^i $(i = 1,\ldots,q,\ j = 1,\ldots,r_i)$ generated at each run of the corresponding optimizer, each set C_j^i is assigned a rank, $rank\left(C_j^i\right)$, equal to one plus the number of sets that are better than it. To state the concept of *better* between two approximation sets, some definitions are necessary. Given two objective vectors z_1 and z_2, z_1 is said to weakly dominate z_2 if z_1 is not worse than z_2 in all objectives (Zitzler et al., 2003). This concept is extended to approximation sets by the set relation *better*. Let A_1 and A_2 be two approximation sets. A_1 is said to be better than A_2 when every objective vector $z_2 \in A_2$ is weakly dominated by at least one $z_1 \in A_1$ and the approximation sets differ from each other, at least, in one objective vector. To verify if an approximation set is better than other, we used the binary additive epsilon indicator (Zitzler et al., 2003). Once all comparisons are performed, ranks are calculated and every optimizer is characterized by a sample $\left(rank\left(C_0^i\right),\ldots,rank\left(C_{r_i}^i\right)\right)$. These samples can be compared using statistical tests. Knowles et al. (2006) claim that, if an optimizer Q^1 has dominance rankings statistically lower than another optimizer Q^2, then it can be said that Q^1 generates better approximation sets than Q^2, and no further testing is required.

In case no statistical difference is observed with the data produced with the dominance ranking method, quality indicators are used. Quality indicators are functions that attribute qualitative values to approximation sets. A unary quality indicator I is defined as a mapping from the set of all approximation sets to the set of real numbers. Some quality indicators use a reference set to compare the approximation sets generated by an optimizer. This reference set can be generated by merging the approximation sets of all optimizers being tested (Knowles et al., 2006). Each indicator measures a specific characteristic of the set and, therefore, combining multiple indicators is suitable. In this work, two of these indicators are used:

1. the hypervolume unary indicator (I_H^-) (Zitzler & Thiele, 1999), which calculates the hypervolume of the objective space that is weakly dominated by the approximation set being tested (limited by a reference point);
2. the additive binary-ϵ indicator $\left(I_{\epsilon+}^1\right)$ (Zitzler et al., 2003), which calculates the minimum ϵ which must be added to each solution in a set A_2 for it to become weakly dominated by another set A_1.

To compare more than two optimizers, we initially use the Kruskal and Wallis (1952) test, and if a statistically significant difference is found at a significance level of 95%, new samples/values are generated for a pairwise comparison. At this stage, two tests are applied: the Wilcoxon (1945) one-tailed test at 97.5% of significance level. If only two optimizers are being compared at a time, there is no need to use Kruskal–Wallis test, and the Wilcoxon test is directly applied. The results returned by the Wilcoxon test are, finally, compared using Taillard, Waelti, and Zuber (2009) test for comparison of proportions with significance level of 95%. For brevity, the only results reported here are the p-values returned by Taillard et al. (2009) test, which are rounded to two decimal places.

All tests were executed on an Intel Xeon QuadCore W3520 2.8 GHz, with 8 GB of RAM running Scientific Linux 5.5 64 bits distribution. Experiments were executed with 18 instances generated by MSPP (2010), from two distinct classes: *square*, which represents a square grid, and *complete*, which contains complete graphs. Fixed processing times for each instance are adopted as stopping criteria. The limits for processing times are set according to each instance size and class. The instances are presented in Table 2 where column # shows the instance identification, column *Type* shows identification ⟨*type*⟩N-⟨*size*⟩ where ⟨*type*⟩ stands for instance type (*Grid*, which represents square instances, or *complete*, which represents complete instances) and ⟨*size*⟩ stands for the instance size (small, medium or large). Moreover, $|N|$, $|A|$ and k are respectively the number of vertices, edges and objectives of each instance, *t(s)* is the runtime (in seconds) used by the variants and S is the seed used for generating the instance. Throughout this work the basic parameters presented in the original GRACE will remain unchanged, that is, $n_{ants} = 300$, $\alpha = 0.6$, $\beta = 0.6$, $\tau_0 = 1$ and $\Delta\tau = \tau_{deposit} = 10$.

5.2. Parameter calibration

For the assessment of all variants, the parameters of each were initially calibrated. Final parameter settings are listed on Table 3. For GRan, 1, 3 and 5 colonies were tested. The single colony version corresponds to the original GRACE. Ants only update the pheromone matrix of its own colony. Results showed the single colony version to outperform the others. A possible explanation for these results is that, since this multi-colony approach is equivalent to a multi-start strategy, using multiple colonies only increases the computational effort.

For PGRan and FUn variants, results showed these variants were competitive or did not improve the current results obtained by GRACE. For PGRan, tests were conducted with $d \in \{1,2,3,4,5\}$, which correspond to 1, 4, 9, 16 and 25 colonies, respectively. The results of the pairwise comparisons showed that the single colony version, which corresponds to the single colony GRan, performs better than all other versions. For the FUn version, values 1, 2 and 3 were tested for parameter d which correspond to 3, 6 and 10 scalarization vectors and colonies. Again, only update by origin is used. No statistically significant differences were found on the complete instances, but for square instances results show that the smaller the value of d, the better the performance of this algorithmic version. It was also noticed that this variant was never able to outperform the original algorithm. For such reasons, both PGRan and FUn were not considered for further experiments.

For the FMu version, tests were conducted using $m \in \{1,3,5\}$ and $d \in \{3,6,10\}$, which correspond to a total of 10, 28 and 66 scalarization vectors, respectively. At first, versions with $m = 1$ were tested for the different values of parameter d. Significant different results were not found either with dominance ranking or the unary quality indicators among those versions. The same occurred for $m = 3$ and 5. This means the m parameter could much likely be tested with any value for d, and results are expected to be similar. Therefore, fixing $d = 3$, comparisons show the same approximation sets have been generated for complete instances, regardless of the configuration used. However, for any number of divisions, the smaller the amount of colonies used, the better the overall performance of the algorithm on square instances. This result is most likely explained by the way weights are assigned to colonies: an assignment according to the order weights are generated does not help colonies specialize in any given region. We also observed that $d = 1$ generates results comparable to $d = 3$. This result is

Table 2
List of instances. Only the first three objectives were used on *large* instances.

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

Table 3
Parameters tested for all variants. The chosen configurations are shown in boldface.

Variant	m	d	Update method
GRan	**1**, 3, 5	–	**Origin**
FUn	d	**1**, 2, 3	**Origin**
FMu	**1**, 3, 5	**3**, 6, 10	**Origin**
PGRan	d^2	**1**, 2, 3, 4, 5	**Origin**
Mix	**1**, 3	**1**, 2, 3	**Origin**
Obj	**3**	**3**, 4, 5	Origin, **region**, both
OMix	**3**	**3**, 4, 5	Origin, **region**, both

interesting, because for $d = 1$ FMu becomes a configuration very similar to FUn, both using only the three canonical weights: (0,0,1), (0,1,0) and (1,0,0). The difference between their performance lies in the fact that FUn uses one colony for each objective, while FMu with $d = 1$ uses a single colony, thus promoting cooperation in the algorithm.

For the Mix variant, results of the previous experiments are used to reduce the number of possible values to be tested for tuning parameters m and d. The values tested for m are 1 and 3, and for d are 3, 6 and 10. The results of the statistical tests concerning pairwise comparisons confirm that the single-colony version prevails on both instance sets (p-$value < 0.01$). The explanation for these results is most likely the same discussed for FMu.

For the tuning of Obj, experiments were conducted initially to set parameter d with testing values 3, 4 and 5. Considering any of the pheromone update methods used in this work (update by origin, update by region and using both strategies at the same time), no significant differences were observed. In the second part of the test, the three pheromone update strategies were directly compared to understand their individual contribution to the performance of the algorithm. Results show that the update by origin strategy is the one that performs worst among the tested square instances. No significant differences were found between the performances of the algorithm that updates by region and the one that uses both strategies simultaneously. This result indicates that the efficiency of the algorithm is directly related to the update by region method.

The same values for parameter d were tested for OMix versions. No significant difference was found for the square instances, regardless of the pheromone update strategy used. For complete instances, however, two different scenarios were found. For all methods, $d = 3$ produced approximation sets significantly better than the versions with $d = 5$. No significant differences were detected between versions with $d = 3$ and $d = 4$. The different update strategies were tested for $d = 3$. No significant difference was found for complete instances, but for squares the same behavior observed for Obj is repeated: the update by region strategy is critical to the efficiency of the algorithm concerning the quality of the approximation sets.

5.3. Results and discussion

The comparison between the variants using dominance rankings shows significant results for almost all pairs of optimizers. Table 4 displays the p-values for the pairwise proportions comparison using the test proposed by Taillard et al. (2009). To make the assessment more straightforward, all values inferior to 0.05 are highlighted in boldface. If the p-value displayed at position (i,j) is lower than 0.05, then variant i is considered to produce better approximation sets than variant j, either according to dominance rankings (column DR) or both dominance rankings and unary quality indicators (column DR + UQI). Hence, if a line i contains no p-values inferior to 0.05, variant i cannot be considered to perform better than any other variant. Also, if any p-value on column i is lower than 0.05, some of the variants were considered to be superior to variant i. Analogously, a high-performing variant can be identified when a given line i contains several p-values lower than 0.05, and the same does not happen on column i.

The observed results are very different for each instance set. Considering only complete instances, Table 4 shows that the GRan variant, which is the original GRACE algorithm, and the Mixed variants are the optimizers that produce the approximation sets with lowest dominance rankings among all optimizers. No difference is found between GRan and Mixed, nor between FMu and Obj. When the quality indicators are assessed, FMu is said to produce approximation sets with lower values in comparison to Obj. For the set of square instances, however, dominance rankings are unable to find significant differences between GRan and Obj, GRan and OMix, FMu and Mix, Mix and Obj, Mix and OMix, and Obj and OMix. Nevertheless, column DR + UQI shows the quality indicators point differences for all such ties. These results point FMu and Obj, which had performed worst on complete instances, to perform best, and both GRan and Mix, which had previously performed best, to perform worst for this set of instances.

Three main conclusions can be drawn from the results of this set of experiments. First, fully stochastic methods such as GRan appear to perform better than systematic approaches like FMu and Obj on unstructured instances, which is the case of the complete graphs. The opposite scenario is observed for highly structured instances such as squares, favoring FMu and Obj. Second, both Mix and OMix variants are unable to maintain dual characteristics, with Mix becoming really similar to GRan, and OMix becoming similar to Obj. Finally, it is interesting to notice that for the MSP using a single colony like in FMu generates better results than using region-specific colonies, like in Obj.

6. Experiments with the novel benchmark

The first set of experiments conducted in this work showed the instance structure to be determinant to the variants performance. In this section, we report an experimentation performed to compare GRan and FMu, the best performing variants on each instance class. A larger benchmark was devised for this purpose, comprising three different graph structures: complete, random and (non-square) grids. Section 6.1 presents the methodology used for the creation, as well as the final instance list used. Section 6.2 describes

Table 4
p-values from pairwise comparisons concerning single-colonies GRan, FMu, and Mix, and 3-colonies Obj and OMix with update by origin and by region simultaneously.

	DR					DR + UQI				
	GRan	FMu	Mix	Obj	OMix	GRan	FMu	Mix	Obj	OMix
Complete										
GRan	–	**0**	0.09	**0**	**0**	–	**0**	0.09	**0**	**0**
FMu	1	–	1	0.5	1	1	–	1	0.5	1
Mix	0.91	**0**	–	**0**	**0**	0.91	**0**	–	**0**	**0**
Obj	1	0.5	1	–	1	1	0.5	1	–	1
OMix	1	**0**	1	**0**	–	1	**0**	1	**0**	–
Square										
GRan	–	0.5	0.5	0.5	0.5	–	1	0.5	1	1
FMu	0.5	–	0.5	0.5	0.5	**0**	–	**0**	**0.04**	**0**
Mix	0.5	0.5	–	0.5	0.5	0.6	1	–	1	1
Obj	0.5	0.5	0.5	–	0.5	**0**	0.96	**0**	–	0.09
OMix	0.5	0.5	0.5	0.5	–	**0**	1	**0**	0.91	–

Table 5
Instances solved using Martins (1984).

Complete			Random			
$\|N\|$	t	$\|P_{set}\|$	$\|N\|$	γ	t	$\|P_{set}\|$
150	0.1043	26.67	1000	18	0.6274	49
180	0.1892	31.67	1000	25	1.1064	43.67
200	0.2883	39.67	5000	18	13.7127	51
240	0.5168	42,67	5000	25	24.5240	59.67
270	0.8085	43	10000	18	53.9591	61.67
300	1.0226	51.34	10000	25	103.9321	60
400	2.2293	66	15000	10	48.6979	40
500	5.3997	67	15000	18	137.4434	64.67
600	10.1386	66.34	15000	25	237.5886	96.34
700	18.2	59,34	20000	25	170.7249	76
800	34.8771	85.34	25000	18	222.6566	71.34
900	52.8754	99.67	25000	25	327.6201	75.34

Table 6
Instance benchmark.

Complete		Random			Grid	
$\|N\|$	t	$\|N\|$	d	t	$\|N\|$	t
1000	46	1000	40	31	20×64	300
1500	168	3000	15	140	20×576	300
2000	300	3000	30	300	60×16	114
2500	300	3000	50	300	60×64	300
3000	300	5000	15	300	60×192	300
4000	300	5000	30	300	80×16	300
5000	300	5000	40	300	80×64	300
6000	300	7000	15	300	150×16	300
7000	300	7000	30	300	150×64	300
8000	300	7000	40	300	300×16	300

the experimental setup for this set of experiments, and the parameter calibration for each variant. Finally, Section 6.3 presents and discuss the results found.

6.1. Benchmark creation

Several instances with different sizes were created, comprising a novel benchmark for performance assessment of optimizers proposed for the MSP. Moreover, to ensure the applicability of metaheuristics, an exact label setting algorithm proposed by Martins (1984) was used to remove from this benchmark instances that could be solved exactly within one hour. Executions were conducted on single cores from an Intel Xeon QuadCore 2.66 GHz with 10 Gb of RAM, running Scientific Linux 5.0. Table 5 shows the average number of non dominated solutions $|P_{set}|$and computational time t found by the algorithm for instances solved within one hour. For each configuration with size $|N|$and $|N| \times \gamma$ edges, three instances were tested.

For complete graphs, Martins' algorithm was able to find the whole Pareto front for instances up to 2000 nodes. When larger instances were considered, the algorithm used all RAM and swap memory, and the process was terminated by the system. For random graphs, instances were initially generated ranging from 1000 to 25000 nodes, with $\gamma \in \{3,6,10,18,25\}$. Martins' algorithm was able to solve most instances within five minutes, and all within one hour. For grid graphs, it was unable to solve the majority of the instances, having solved only 20×16, 20×64, and 60×16 nodes instances.

The filtered benchmark contains 10 different configurations per class ranging from 1000 to 8000 nodes. For each configuration, 10 instances were generated with different seeds, totalizing 300 instances. Details can be seen on Table 6. The d column for random instances stands for the graph density. For the time limits in seconds t, we used $min\{t_{exact}$,300}, where t_{exact} is the time used by the exact algorithm for the instances it was able to solve.

6.2. Methodology and parameter calibration

For the calibration of the variants, a representative sample was chosen from the benchmark (one instance per configuration). Fifteen runs were performed by each optimizer for each parameter setting during the calibration stage. All parameters tested are listed on Table 7, and the final settings are highlighted in boldface. For the comparison of the variants, each algorithm performed three independent runs on the remaining 270 instances, totaling 810 runs per algorithm.

The assessment of the optimizers was conducted using the same methodology from the first set of experiments. This time, however, each algorithm was executed on a single core from an Intel Xeon QuadCore 2.66 GHz with 10 Gb of RAM. Since only three executions were conducted per instance, we used the original Friedman test (Conover, 1999), considering the average between the three runs.

6.3. Results and discussion

The Friedman test results on the dominance rankings produced by both optimizers show FMu to generate approximation sets with statistically lower rankings than GRan when all instances are considered simultaneously. The same result holds when instance classes are considered separately, except for grid instances, where no significant difference among them is found. Further investigation was conducted on the quality indicators for this set, and again FMu produced sets with lower values for both indicators, at significance level 0.05. Tables 8 and 9 show the average values per configuration of each indicator.

For complete instances, FMu is able to generate better approximation sets than GRan for instances with up to 3000 nodes, and the opposite happens for larger instances. For grid instances, the $I^1_{\epsilon+}$ indicator does not let us draw conclusions on why GRan and FMu interchangeably produce good results, but the I^-_H shows FMu to generate approximation sets with lower values than GRan for all instances but the largest two. For random instances, FMu generates better values than GRan for all instances but the smaller two according to both indicators. For those two instances the $I^1_{\epsilon+}$

Table 7
Parameters tested for GRan and FMu.

Variant	#Ants	α	β	τ_0	$\Delta\tau$
GRan	150, 300, 500	0.4, 0.6, 0.8	0.5, 0.7, 0.9	0.5, 1, 10	5, 10, 15
FMu	150, 300, 500	0.4, 0.6, 0.8	0.5, 0.7, 0.9	0.5, 1, 10	5, 10, 15
	ρ	m	d	Update method	
GRan	0, 0.1, 0.3	1, 3, 5	-	Origin	
FMu	0, 0.1, 0.3	1, 3, 5	1, 3, 5	Origin	

Table 8
Indicators from complete/grid instances.

$\|N\|$	$I^1_{\epsilon+}$		I^-_H	
	GRan	*FMu*	*GRan*	*FMu*
Complete				
1000	1415.83	1346.10	1.42e−11	1.20e−11
1500	1415.83	1346.10	1.42e−11	1.20e−11
2000	1180.80	1118.83	1.48e−11	1.15e−11
2500	1323.87	1285.80	1.70e−11	1.37e−11
3000	1099.83	1090.93	1.51e−11	1.36e−11
4000	1039.20	1149.27	1.10e−11	1.29e−11
5000	1051.80	1057.17	1.14e−11	1.06e−11
6000	642.67	754.20	6.48e−10	7.03e−10
7000	670.40	715.93	5.64e−10	7.23e−10
8000	663.03	635.17	5.36e−10	5.40e−10
Grid				
20 × 64	6185.57	6267.60	1.01e+14	9.90e+13
20 × 576	6185.57	6267.60	1.01e+14	9.90e+13
60 × 16	18785.03	18438.47	3.45e+14	3.38e+14
60 × 64	5638.07	5678.57	4.27e+13	4.25e+13
60 × 192	11170.16	10612.20	1.24e+14	1.23e+13
80 × 16	10132.67	10302.33	6.92e+13	6.53e+13
80 × 64	8411.03	8266.83	9.24e+13	9.19e+13
150 × 16	9436.03	9519.23	1.53e+14	1.50e+14
150 × 64	12857.07	12923.30	1.63e+14	1.64e+14
300 × 16	14828.33	13942.73	3.57e+14	3.72e+14

Table 9
Indicators from random instances.

Random		$I^1_{\epsilon+}$		I^-_H	
$\|N\|$	*d*	*GRan*	*FMu*	*GRan*	*FMu*
1000	40	1298.80	1330.10	1.59e−11	1.35e−11
3000	15	1298.80	1330.10	1.59e−11	1.35e−11
3000	30	1389.33	1359.37	1.40e−11	1.29e−11
3000	50	1393.63	1350.13	1.59e−11	1.18e−11
5000	15	1247.93	1233.77	2.35e−11	2.01e−11
5000	30	1285.30	1250.13	1.87e−11	1.45e−11
5000	40	1000.73	871.33	1.44e−11	1.08e−11
7000	15	1034.70	921.30	7.92e−10	7.32e−10
7000	30	1205.77	1169.07	1.53e−11	1.09e−11
7000	40	839.23	740.10	1.40e−11	1.14e−11

indicator favors GRan, whether the I^-_H indicator favors FMu, which means the approximation sets are most likely incomparable.

These results show that the different underlying graph structures become less important as instances grow larger. One possible explanation is that for a larger search space systematic approaches tend to degenerate less than stochastic methods, favoring FMu over GRan. Since this configuration of GRan is equal to the original GRACE algorithm, these results show the FMU variant proposed in this paper is able to generate approximation sets of better quality than the state-of-the-art of the problem just by changing the methods for the generation and assignment of scalarization vectors.

7. Conclusions and future work

This paper addressed the importance of studying the impact of different algorithmic components for MOACO algorithms when designing an optimizer for a particular MCOP. Particularly, a MOACO algorithm named GRACE from the state-of-the-art of the multi-objective shortest path was extended and improved. Moreover, different methods for the generation and assignment of scalarization vectors for MOACO algorithms were proposed and compared, which can possibly be extended to multi-objective metaheuristics in general. Finally, a new instance benchmark was proposed for the MSP, which may be used from now onto compare different optimizers proposed for this problem.

As future work possibilities, the methods for generating and assigning weights to colonies should be studied for different MCOPs, in order to test the generality of the results found in this work. Moreover, the same topic addressed in this paper for MOACO algorithms needs to be extended to other multi-objective metaheuristics, such as GRASP, Particle Swarm Optimization and EMO algorithms such as the MOEA/D (Zhang & Li, 2007).

References

Alaya, I., Solnon, C., & Ghédira, K. (2007). Ant colony optimization for multi-objective optimization problems. In *ICTAI'07* (pp. 450–457).

Barán, B., & Schaerer, M. (2003). A multiobjective ant colony system for vehicle routing problem with time windows. In *IASTED'03* (Vol. 21, pp. 97–102).

Bezerra, L. C. T., Goldbarg, E. F. G., Goldbarg, M. C., & Buriol, L. S. (2011). Grace: A generational randomized aco for the multi-objective shortest path problem. In R. Takahashi, K. Deb, E. Wanner, & S. Grecco (Eds.), *EMO'11. LNCS* (Vol. 6576, pp. 535–549). Springer.

Chica, M., Cordón, O., Damas, S., & Bautista, J. (2011). A new diversity induction mechanism for a multi-objective ant colony algorithm to solve a real-world time and space assembly line balancing problem. *Memetic Computing, 3*, 15–24.

Conover, W. J. (1999). *Practical non-parametric statistics* (3rd ed.). New York, NY: John Wiley and Sons.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6*, 182–197.

Dijkstra, E.W. (1959). A note on two problems in connexion with graphs. Nr. Math.

Doerner, K., Gutjahr, W., Hartl, R., Strauss, C., & Stummer, C. (2004). Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection. *Annals of Operations Research, 131*, 79–99.

Doerner, K., Gutjahr, W., Hartl, R., Strauss, C., & Stummer, C. (2006). Pareto ant colony optimization with ILP preprocessing in multiobjective project portfolio selection. *European Journal of Operational Research, 171*, 830–841.

Doerner, K., Hartl, R. F., & Reimann, M. (2001). Are competants more competent for problem solving?: - The case of a multiple objective transportation problem. *GECCO'01*. Berlin, Heidelberg: Morgan Kaufmann, p. 802.

Dorigo, M. (1992). Optimization, learning and natural algorithms. Ph.D. thesis, Dipartamento de Elettronica, Politecnico de Milano.

Dorigo, M., & Socha, K. (2006). An introduction to ant colony optimization. Technical report, TR/IRIDIA/2006-010 IRIDIA, Université Libre de Bruxelles.

Dorigo, M., & Gambardella, L. M. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation, 1*, 53–66.

García-Martínez, C., Cordón, O., & Herrera, F. (2007). A taxonomy and empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. *European Journal of Operational Research, 180*, 116–148.

Ghoseiri, K., & Nadjari, B. (2009). An ant colony optimization algorithm for the bi-objective shortest path problem. *Applied Soft Computing, 10*, 1237–1246.

Grass é, P.-P. (1959). La reconstruction du nid et les coordinations interindividuelles chez bellicositermes natalensis et cubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs. *Insectes Sociaux, 6*, 41.

Häckel, S., Fischer, M., Zechel, D., & Teich, T. (2008). A multi-objective ant colony approach for pareto-optimization using dynamic programming. *GECCO'08*. ACM, pp. 33–40.

He, F., Qi, H., & Fan, Q. (2007). An evolutionary algorithm for the multi-objective shortest path problem. In *ISKE'07*.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. University of Michigan Press.

Iredi, S., Merkle, D., & Middendorf, M. (2001). Bi-criterion optimization with multicolony ant algorithms. In E. Zitzler, L. Thiele, K. Deb, C. A. C. Coello, & D. Corne (Eds.), *EMO. LNCS* (Vol. 1993, pp. 359–372). Springer.

Ke, L., Feng, Z., Xu, Z., Shang, K., & Wang, Y. (2010). A multiobjective aco algorithm for rough feature selection. In *PACCS'10* (pp. 207–210).

Knowles, J. D., Thiele, L., & Zitzler, E. (2006). A tutorial on the performance assessment of stochastic multiobjective optimizers. Technical report, TIK-Report No. 214.

Kruskal, W. H., & Wallis, W. A. (1952). Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association, 47*, 583–621.

López-Ibáñez, M., Paquete, L., & Stützle, T. (2004). On the design of ACO for the biobjective quadratic assignment problem. In M. Dorigo, M. Birattari, C. Blum, L. Gambardella, F. Montada, & T. Stützle (Eds.), *ANTS. LNCS* (Vol. 3172). Springer-Verlag.

López-Ibáñez, M., & Stützle, T. (2010). The impact of design choices of multiobjective ant colony optimization algorithms on performance: An experimental study on the biobjective TSP. *GECCO '10*. ACM, pp. 71–78.

Martins, E. Q. V. (1984). On a multicritera shortest path problem. *European Journal of Operational Research, 16*, 236–245.

Mooney, P., & Winstansley, A. (2006). An evolutionary algorithm for multicriteria path optimization problems. *International Journal of Geographical Information Science, 20*, 401–423.

Mora, A. M., Merelo, J. J., Laredo, J. L. J., Millan, C., & Torrecillas, J. (2009). Chac, a moaco algorithm for computation of bi-criteria military unit path in the battlefield: Presentation and first results. *International Journal of Intelligence Systems, 24*, 818–843.

Mote, J., Murphy, I., & Olson, D. L. (1991). A parametric approach to solving bicriterion shortest path problems. *European Journal of Operational Research, 53*, 81–92.

MSPP. (2010). <www.mat.uc.pt/∼zeluis/INVESTIG/MSPP/mspp.htm>.

Pangilinan, J. M. A., & Janseens, G. K. (2007). Evolutionary algorithms for the multiobjective shortest path problem. *International Journal of Computing and Information Science in Engineering, 1*.

Paquete, L., & Stützle, T. (2003). A two-phase local search for the biobjective traveling salesman problem. In C. Fonseca, P. Fleming, E. Zitzler, L. Thiele, & K. Deb (Eds.), *EMO. LNCS* (Vol. 2632, pp. 69). Berlin/Heidelberg: Springer.

Raith, A., & Ehrgott, M. (2009). A comparison of solution strategies for biobjective shortest path problems. *Computers & Operations Research, 36*, 1299–1331.

Skriver, A. J. V., & Andersen, K. A. (2000). A label correcting approach for solving bicriterion shortest-path problems. *Computers & Operations Research, 27*, 507–524.

Steiner, S., & Radzik, T. (2003). Solving the biobjective minimum spanning tree problem using a *k*-best algorithm. Technical report, TR-03-06 Department of Computer Science, King's College London.

Stützle, T., & Hoos, H. H. (2000). Max-min ant system. *Future Generation Computer Systems, 16*, 889–914.

Taillard, E., Waelti, P., & Zuber, J. (2009). Few statistical tests for proportions comparison. *European Journal of Operational Research, 185*, 1336–1350.

Vianna, D. S., & Arroyo, J. E. C. (2004). A GRASP algorithm for the multi-objective knapsack problem. In *XXIV International Conference of the Chilean Computer Science Society*.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometric Bulletin, 1*, 80–83.

Zhang, Q., & Li, H. (2007). Moead: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation, 11*, 712–731.

Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength pareto evolutionary algorithm. *IEEE Transactions on Evolutionary Computation, 3*, 257–271.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., & da Fonseca, V. G. (2003). Performance assessment of multiobjective optimizaers: An analysis and review. *IEEE Transactions on Evolutionary Computation, 7*, 132.