Multi-Objective Ant Colony Optimization

Manuel López-Ibáñez Infante

Supervisors:

Luis Paquete **Thomas Stützle**

October 2003 – July 2004

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Universidad de
Granada

License

Preface

Real world problems usually involve several and often conflictive objectives that must be simultaneously optimized in order to achieve a satisfactory solution. Multi-objective optimization has its roots at the end of the nineteenth century, in the studies of Edgeworth and Pareto, but had not experimented a great development until the fifties. Since last decade, there is an ongoing research effort to develop approximate algorithms and metaheuristic approaches to solve multi-objective problems. As a result, various metaheuristic approaches have been applied to multi-objective optimization, including simulated annealing (SA), taboo search (TS) and evolutionary algorithms. Nevertheless, there are many important open questions in multi-objective optimization, e.g., the adequate performance assessment of multi-objective optimization algorithms.

Ant Colony Optimization (ACO) is a metaheuristic approach applied successfully to single objective combinatorial problems, but few work has been done to apply ACO principles to multi-objective combinatorial optimization (MOCO) and most of the proposed algorithms are only applicable to problems where the objectives can be ordered according to their importance. Therefore, it is still unclear how ACO algorithms for these problems should be designed.

This work examines the concepts involved on the design of ACO algorithms to tackle MOCO problems, e.g., definition of multi-objective pheromone information, multiple cooperative ant colonies, pheromone update strategies in the multi-objective context and local search methods for multiple objectives. Most of these concepts have been previously used in the literature. Nevertheless, we examine them independently of any particular problem and we propose alternative concepts still not considered. Moreover, we study these concepts in relation to the available knowledge about the best performing ACO algorithms for single objective optimization and the search strategies currently used in multi-objective optimization. As well, these concepts are studied as components of a general multi-objective ACO (MO-ACO) algorithm.

Next, we apply these concepts to design an ACO algorithm for the bi-objective case of a recently proposed multi-objective variant of the quadratic assignment problem (QAP). Therefore, we consider various configurations of the MO-ACO algorithm and compare their performance when applied to the bi-objective QAP. These configurations use the $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System as the underlying algorithm because it is considered the best performing ACO algorithm for the single objective QAP. In addition, the particular components of each configuration are chosen in such a way that the configuration reflects a particular search strategy. This search strategy is either based on dominance criteria or based on scalarizations of the objective vector. To address the performance assessment of these experiments, we use a methodology that combines up-to-date ideas from the research area of multi-objective optimization with well-known statistical techniques from experimental design.

The experimental results show that the use of local search with MO-ACO is essential for the bi-objective QAP. Moreover, other parameters become less significant when local search is applied. The underlying search strategy followed by each configuration plays an important role in the shape of the Pareto set obtained by the algorithm.

The organization of concepts of MO-ACO into modular components which can be combined into various configurations allows applying MO-ACO to other MOCO problems. The available knowledge of single objective ACO, multi-objective optimization and about particular problems may be used to design new components, e.g., different local search methods, which can be combined into configurations of MO-ACO in order to tackle different MOCO problems.

This document is structured as follows:

Chapter 1 introduces the basic concepts of multi-objective optimization in terms of Pareto optimality.

Chapter 2 reviews the single objective QAP, describing the problem definition and briefly the current knowledge about types of QAP instances and related measures. Afterward, this chapter introduces the proposals on multi-objective QAP, and finally it describes the bi-objective QAP instances used as test instances.

Chapter 3 gives a brief introduction to Ant Colony Optimization, the application of ACO to the single objective QAP and the $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System.

Chapter 4 examines the design of multi-objective ACO algorithms. Particularly, we discuss the use of multiple colonies, several pheromone update strategies and different search strategies. Finally, several configurations are considered in order to tackle the bQAP.

Chapter 5 describes the assessment methodology followed to evaluate the configurations considered to tackle the bQAP.

Chapter 6 analyzes the results of the experiments performed with respect to the assessment methodology.

Chapter 7 concludes and indicates future research directions.

Contents

List of Figures

List of Tables

Chapter 1

Multi-objective Optimization

Many optimization problems arising in practice involve more than one optimization criterion, requiring the simultaneous optimization of several, perhaps conflicting, objective functions. Hence, a single optimal solution is not as interesting as information about the trade-off between the various objectives. In multi-objective optimization, the abstract concept of trade-off is usually defined in terms of Pareto optimality. Thus, the goal is to obtain the set of all Pareto optimal solutions or, at least, a good approximation of this set. An alternative method would be to transform the multi-objective problem into a single objective problem using a weighted sum of the multiple objectives. Then, Pareto optimal solutions can be found by solving several of such single objective problems using various weights. However, if certain conditions are not met, not all Pareto optimal solutions will be found by means of such method. In this work we are only interested in a certain class of multi-objective problems, multi-objective combinatorial optimization problems.

1.1 Multi-objective Optimization Problems

As its very name suggests, in a multi-objective optimization problem (also called multi-criteria optimization and vector optimization problem) every solution is measured according to more than one objective function, each of which must be minimized or maximized. Let Q be the number of objectives in a multi-objective problem, let $\mathcal{S}$ be the set of feasible solutions and let Z denote the set of all objective value vectors of feasible solutions. Each element of Z is a vector referred to as *objective vector* formed by the values of the objective functions. Formally,

$$\forall s \in \mathcal{S}, \quad \exists z \in Z, \quad z = (z_1, \dots z_Q)\colon \begin{cases} z_1 = f_1(s) \\ \quad\vdots \\ z_Q = f_Q(s) \end{cases}$$

The feasible set $\mathcal{S}$ is a subset of a space called the *search space*. The space from which objective values are taken is called the *objective space*.

Let us assume that all Q objectives should be minimized, without loss of generality. Thus, the goal of multi-objective optimization is to solve

$$\min_{s \in \mathcal{S}} (f_1(s), \ldots, f_Q(s)) \tag{1.1}$$

where the meaning of “min”, i.e., the concept of optimality in multi-objective optimization, will be defined in the following section.

1.2 Pareto Optimality

Given two objective vectors $u, v \in Z : u = (u_1, \ldots, u_Q),\ v = (v_1, \ldots, v_Q)$, we need to define whether u is better than v in order to find an optimal objective vector which is the solution to the multi-objective problem described above. When objectives can be ordered according to their importance, optimality can be defined in terms of lexicographic order. Let us suppose objectives are arranged in decreasing order of importance, i.e., $q = 1$ is the most important objective and $q = Q$ is the least important one. Then, $u <_{lex} v$ if $u_q < v_q$, where q is the smallest index such that $u_q \neq v_q$.

However, in the most general case, we cannot assume *a priori* that there is a ranking among the objectives. Then, the only assumption we can make is that u is better than or preferable to v, if u is not worse than v in all objectives and better with respect to at least one objective. This relation is commonly known as the concept of Pareto dominance.

More formally, we say that u dominates v ($u \prec v$) if $u \neq v \wedge u_q \leqslant v_q$, $q = 1, \ldots, Q$. From this definition, if $u \prec v$ then $v \nprec u$. When $u \nprec v \ \wedge\ v \nprec u \ \wedge\ u \neq v$, we say that u and v are nondominated vectors ($u \parallel v$).[1]

For simplification, given two solutions $s, r \in \mathcal{S}$ and their respective objective vectors $u, v \in Z : u = \big(f_1(s), \ldots, f_Q(s)\big),\ v = \big(f_1(r), \ldots, f_Q(r)\big)$, we will say that s dominates r if and only if u dominates v. Therefore, in the following, when referring to dominance relations between solutions, one means that relations are applied in the objective space, i.e., to the objective vectors that those solutions represent.

The Pareto dominance relation induces a partial order on the search space such that we can define a *Pareto optimal* solution which is not dominated by any other feasible solution. However, several such solutions may exist as nondominated objective vectors, thus forming a set of nondominated Pareto

[1] The notation used assumes a minimization problem. However, the notation found in the literature (e.g., [58]), could have a different meaning. As well exposed by Papadimitriou & Yannakakis [41]: “The dichotomy of maximization vs. minimization is a well-known complication that is usually technically inconsequential, but burdens the exposition and notations. [...] with multi-objective problems, in which each objective can be independently either a maximization or minimization problem, the situation is *exponentially* more complicated.” For the sake of clarity, we will never use a notation like $u \succ v$ to show that v dominates u. This situation will be noted as $v \prec u$. Consequently, when u dominates v, we will denoted it as $u \prec v$. This rule will be followed in all the relations used in this work.

optimal objective vectors. The set containing all the Pareto optimal solutions is called *optimal Pareto set.* In a more general sense, any set of nondominated objective vectors, regardless of their optimality, will be called *Pareto set.* In the literature, the Pareto set is sometimes called Pareto front, curve and surface, since the image in the objective space of a Pareto set resembles a frontier or surface.

Much of the current research on the field of multi-objective optimization is concerned with the problem of how to identify the optimal Pareto set or, at least, to produce Pareto sets which are good approximations of it. Thus, the question that arises is how to describe whether a Pareto set is a good approximation to the Pareto optimal set, that is, how to describe the *quality* of a Pareto set.

1.3 Relations between Pareto Sets

The quality of a Pareto set, in general, cannot be completely described by means of a numeric criterion, i.e., a unary measure that assigns a numeric value to a given Pareto set [58]. However, we can completely describe the quality of Pareto sets when they are compared with other Pareto sets.

Let A and B, $A \neq B$, be two Pareto sets which may be the respective outcomes for a particular problem instance of two optimization algorithms that we want to compare. First, we can say that A is *better* than B ($A \lhd B$) if any objective vector in B is dominated by or equal to at least one objective vector in A. Formally, it is defined as

$$A \lhd B \iff \forall b \in B, \exists a \in A : \quad a \prec b \ \vee \ a = b$$

This relation represents the most general and weakest form of higher quality between Pareto sets. It is also a transitive relation, i.e., given the Pareto sets A, B and C, if $A \lhd B$ and $B \lhd C$ then $A \lhd C$.

When neither $A \lhd B$ nor $B \lhd A$, we say that the two sets are incomparable ($A \parallel B$) in terms of Pareto dominance. This means that using only the Pareto optimality criterion, we cannot state that one of these Pareto sets is preferable to the other. Yet, they are not equal, thus there could be other criteria to prefer one of them.

Finally, we can say that A is not worse than B, or in other words, that B is not better than A, ($B \ntriangleleft A$). Formally, $B \ntriangleleft A \iff A \lhd B \ \vee \ A \parallel B$. This relation is interesting because existing comparison methods based on unary measures at best allow to indicate whether a Pareto set is not worse than another [58].

How to choose between Pareto sets that are incomparable in terms of Pareto optimality is still an open question in multi-objective optimization, since several vague criteria can be taken into account such as the number of solutions and the spread distribution in the objective space.

1.4 Weighted Sum Scalarization

Any multi-objective optimization problem can be transformed into a single objective scalarized problem of the type

$$\min_{s \in S} \sum_{q=1}^{Q} \lambda_q f_q(s)$$

where λ_q is the q^{th} component of a weight vector λ taken from the set of weight vectors as

$$\Lambda = \left\{ \lambda \in \mathbb{R}^Q : \lambda_q \geq 0, \sum_{q=1}^{Q} \lambda_q = 1 \right\} \tag{1.2}$$

Optimal solutions for scalarized problems where all the weights are positive ($\lambda_q > 0$) are always Pareto optimal, and under certain assumptions all Pareto optimal solutions are optimal solutions of scalarized problems with nonnegative weights [15]. When these assumptions are not met, all the Pareto optimal solutions may not be found by solving several scalarized problems [15]. Nevertheless, a part of the current research on multi-objective optimization is focused on search strategies based on solving several scalarizations of the objective vector by means of weighted sum.

1.5 Multi-objective Combinatorial Optimization Problems

The fundamental property of combinatorial optimization problems is that they have a finite set of feasible solutions, and thus the objective space is discrete, as opposed to continuous problems.

Let E be a finite set $E = \{e_1, \ldots, e_n\}$ and let $w \colon E \to \mathbb{Z}$ be a weight function[2] on the elements of E. Then, a finite set of constraints Ω defines the feasible set $\mathcal{S}$ of a combinatorial problem, with $\mathcal{S} \subset 2^E$,

Finally, given an objective function $f \colon \mathcal{S} \to \mathbb{R}$ of the type of $f(s) = \sum_{x \in s} w(x)$ or $f(s) = \max_{x \in s} w(x)$ or even a more complex function, then a combinatorial optimization problem (f, Ω) is formulated as

$$\min_{s \in \mathcal{S}} f(s) \tag{1.3}$$

An *instance* of a combinatorial optimization problem is given by a specific set E and a specific weight function w, usually given as a vector or a matrix of size $|E| = n$. For this reason, n is called the *size* of the instance.

[2] This weight function should not be confused with the weight vector described in the previous section. They are different concepts without any relationship.

Multi-objective combinatorial optimization (MOCO) deals with combinatorial problems with Q objective functions. Hence, the above formulation (1.3) of a combinatorial problem is extended to a multi-objective variant similar to Equation (1.1).

With regard to the difficulty of MOCO problems, there are three important aspects that should be considered [16]. Firstly, the size of the optimal Pareto set may be exponential with respect to the instance size, thus there may not be any computational method to calculate the whole optimal Pareto set in a reasonable time. Secondly, because of the discrete structure of MOCO problems, there usually exist Pareto optimal solutions, called nonsupported solutions, which are not optimal for any weighted sum of the objectives. Furthermore, there may be many more nonsupported than supported Pareto optimal solutions [56]. Lastly, even in the case of single objective problems which are solvable in polynomial time, the respective MOCO versions may be $\mathcal{NP}$-hard.

Several MOCO problems have been tackled in the literature by means of multi-objective versions of the shortest path, knapsack and traveling salesperson problems, to mention only a few (see [16] for a complete survey). In this work, we will use as a test problem a recently proposed multi-objective formulation of the quadratic assignment problem.

Chapter 2

The Quadratic Assignment Problem

Much of the current research on multi-objective combinatorial optimization (MOCO) is based on the vast knowledge and literature of single objective problems. The quadratic assignment problem (QAP) is an interesting benchmark problem and it has been broadly studied, since it has a very simple formulation which does not need specialized heuristics and/or repair mechanisms, the objective function is fast to compute and it can also be delta-evaluated, enabling local search to be efficiently applied. Furthermore, the QAP is both practically important and unusually difficult. Recently, multi-objective variants of the QAP have been proposed. The present work will focus on structured and unstructured instances of the bi-objective QAP.

2.1 The Single Objective QAP

The *quadratic assignment problem* (QAP) was introduced by Koopmans & Beckmann [34] in 1957, becoming an important problem from the point of view of practice and theory. The QAP is usually described as the problem of assigning a set of facilities to a set of locations with given distances between the locations and given flows between the facilities. The goal is to place the facilities on locations in such a way that the sum of the products between flows and distances is minimal.

More formally, given the set of integers $N = \{1, 2, \ldots, n\}$ and two $n \times n$ matrices $A = [a_{ij}]$ and $B = [b_{rs}]$, the QAP is stated as follows:

$$\min_{\phi \in \Phi} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b_{\phi_i \phi_j} \tag{2.1}$$

where Φ is the set of all possible permutations of N.

In the usual description of the problem, a_{ij} is the distance between locations i and j, b_{rs} is the flow between facilities r and s, and ϕ_k gives the facility assigned to the location k in the current solution.

Many different practical planning problems can be formulated as QAP instances: backboard wiring [46], campus and hospital layout [7, 18], typewriter keyboard design [4], scheduling [24], and many others [17, 35].

From the theory point of view, the QAP is an $\mathcal{NP}$-hard combinatorial optimization problem and even finding a solution within a factor of $1 + \epsilon$ of the optimal one remains $\mathcal{NP}$-hard [45]. Even relatively small instances ($n \geqslant 30$) cannot be solved to optimality. It is considered as one of the hardest optimization problems, which has lead into a great research effort in approximate methods. Moreover, the QAP is a broad problem class embracing both the *graph partitioning problem* and *traveling salesman problem* as special cases [40].

2.1.1 Types of QAP Instances

The existence of various types of QAP instances and the influence of the particular instance type on the performance of heuristic methods are widely known [51, 53]. The best known classification differences between four classes of instances:

(*i*) **Unstructured, randomly generated, instances.** Instances with the distance and flow matrix entries generated randomly according to a uniform distribution.

(*ii*) **Grid-based distance matrix.** In this class of instances the distance matrix stems from a $n_1 \times n_2$ grid and the distances are defined as the Manhattan distance between grid points. These instances have multiple global optima (at least 4 if $n_1 \neq n_2$ and at least 8 if $n_1 = n_2$) due to the definition of the distance matrices.

(*iii*) **Real-life instances.** These instances have been obtained from practical applications of the QAP. Real-life instances have in common that the flow matrices have many off-diagonal zero entries and the remaining entries are clearly not uniformly distributed.

(*iv*) **Structured, real-life-like instances.** Since the real-life instances are of a rather small size, a particular type of randomly generated problems has been proposed in [53]. These instances are generated in such a way that the matrix entries resemble the distributions found for real-life problems.

In the remainder of this work, we will only deal with unstructured and structured instances.

Structured instances stemming from applications are practically more interesting than unstructured ones. In terms of difficulty, although finding the optimum (or best known solution) is easier on structured instances of small size, these instances are generally more difficult in terms of finding solutions within a given percentage from the optimum. In contrast, in the unstructured instances there are far more solutions at a given percentage deviation from the optimum and finding which of these is close in the search space to the optimum is then difficult. Therefore, unstructured instances are among the hardest to solve exactly, although most iterative search methods find solutions within 1 – 3% from the best known solutions relatively fast [53].

2.1.2 Measures of QAP Instances

Some computationally tractable properties of the distance and flow matrices are sufficient to *characterize* an instance as belonging to one of the above classes.

In general, the dominance statistic $dom(X)$ of a matrix $X = [x_{ij}]$ is defined as:

$$dom(X) = 100 \cdot \frac{\sigma}{\mu} \qquad \text{where}$$

$$\mu = \frac{1}{n^2} \cdot \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} \qquad \text{and} \qquad \sigma = \sqrt{\frac{1}{n^2 - 1} \cdot \sum_{i=1}^{n} \sum_{j=1}^{n} (x_{ij} - \mu)}$$

Using the above definition, Vollmann & Buffa [57] introduced the *flow dominance* as the dominance of the flow matrix, $fd{=}dom(B)$. A high flow dominance indicates that a large part of the overall flow is exchanged among relatively few items. In an analogous way, the distance dominance dd can be defined on the distance matrix A.

Unstructured instances have a rather low flow dominance, whereas structured ones, in general, have a rather high flow dominance. Furthermore, structured instances often have many zero entries in the flow matrix. Hence, the *sparsity* of the flow matrix, defined as $sp = n_0/n^2$ where n_0 is the number of zero matrix entries, can give additional information about the type of the instance.

2.2 The Multi-objective QAP (mQAP)

The single objective QAP can be extended to a multi-objective variant which models the situation where amounts of different goods are exchanged between facilities (as modeled by different flow matrices) and the exchange can be done with different speeds (as modeled by different distance matrices).

Two proposed definitions for the multi-objective QAP (mQAP) can be found in the literature. Hamacher *et al.* [28] consider several matrices of flows and distances for modeling problems found in facility layouts for social institutions. Knowles & Corne [32] proposed a different variant using multiple flow matrices and keeping the same distance matrix. They give an example of a practical application in hospital layout problems where the flows are doctors, patients, visitors and products.

The mQAP proposed by Knowles & Corne is formalized as

$$\min_{\phi \in \Phi} \begin{cases} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b^{1}_{\phi_i \phi_j} \\ \vdots \\ \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b^{Q}_{\phi_i \phi_j} \end{cases} \tag{2.2}$$

where Q is the number of objectives, b^q_{rs} for $q = 1, \ldots, Q$ is the flow matrix entry corresponding to the q^{th} flow matrix and "min" is understood in terms of Pareto optimality.

2.2.1 Bi-objective QAP (bQAP) Instances

In this work we focus on the bi-objective version of the proposal of Knowles & Corne, i.e., the model in (2.2) with $Q = 2$. All the instances were obtained using the generators[1] for unstructured and structured instances provided by Knowles & Corne [32]. Both generators produce instances with one distance matrix and multiple flow matrices, all of them are *symmetric* matrices.

Unstructured instances have been previously generated by Luis Paquete[2] using the uniformly random instance generator, resulting in symmetric bi-objective QAP instances where the matrix entries of all matrices are drawn randomly according to a uniform distribution in the interval $[1, 100]$ and flow matrices are generated with a correlation ξ which induces certain correlation between the two objectives.

Structured instances have been generated using the real-life like instance generator. In these instances, the distance matrix corresponds to the Euclidean distance between n points on a plane. The program generates clusters of a number of points in $[1, K]$ which are uniformly distributed in a circle of radius M and the points in a circle of radius m. The flow matrices are also randomly generated following a distribution with parameters A and B. Using parameters $M = 1000$, $K = 20$, $m = 10$, $A = -10$ and $B = 5$, the instances generated resemble single objective QAP instances named `Taixxb` in [53]. With $A = -10$ and $B = 5$ the flow matrices contain about 2/3 of zero entries and the maximum value of the flows is 10^5. A correlation between the objectives is induced by generating flow matrices with a certain correlation ξ. Finally, an overlap parameter η indicates the fraction of entries in the

[1] Source code is available at `http://iridia.ulb.ac.be/~jknowles/mQAP/gens.html`

[2] These instances are available at `http://www.intellektik.informatik.tu-darmstadt.de/~lpaquete/QAP/`

second flow matrix that are correlated with the corresponding entries in the first flow matrix. As explained by Knowles & Corne [32], when $\eta = 0$, a random uncorrelated value is placed in each entry of the second flow matrix that corresponds to a zero entry in the first flow matrix. Conversely, a zero is placed in each entry of the second flow matrix that corresponds to a non-zero value in the first flow matrix. Thus, there is no overlap between the flows of the first and second matrices when $\eta = 0$. We generated all the structured instances using $\eta = 1$, thus all the flows overlap and are correlated.

The parameters and properties of instances used in this work are summarized in Table 2.1 for the unstructured instances and in Table 2.2 for the structured ones.

Instance	`qapUni.50.0.1`	`qapUni.50.n75.1`	`qapUni.50.p75.1`
$\max(d)$	100	100	100
$\max(f)$	100	100	100
$\xi(f^1, f^2)$	0	-0.75	0.75
fd^1, fd^2	59.7, 59.1	58.5, 57.0	58.9, 56.8
dd	58.2	59.5	61.1
sp^1, sp^2	0.02, 0.02	0.02, 0.02	0.02, 0.02
Seed	635708	574477	550710

Table 2.1: *b*QAP unstructured instances. The parameters/properties are: $\max(d)$, the maximum distance in the distance matrix; $\max(f)$, the maximum flow in any of the flow matrices. $\xi(f^1, f^2)$, the correlation parameter affecting corresponding flow matrix entries of the flow matrices; fd^i, the flow dominance of the ith flow matrix; *dd*, the distance dominance; sp^i, sparsity of the ith flow matrix. The random *seed* to the generator is also given for reference.

Instance	`qapStr.50.0.1`	`qapStr.50.n75.1`	`qapStr.50.p75.1`
$\max(d)$	1539	1539	1539
$\max(f)$	98787	99860	98091
$\xi(f^1, f^2)$	0	-0.75	0.75
fd^1, fd^2	357.7, 399.5	425.3, 433.6	425.3, 459.9
dd	67.9	67.9	67.9
sp^1, sp^2	0.6536, 0.6776	0.6752, 0.6736	0.6752, 0.6664
Seed $= 23453464$, $A = -10$, $B = 5$, $M = 1000$, $K = 20$, $m = 10$, $\eta = 1.0$			

Table 2.2: *b*QAP structured instances. The parameters/properties are: $\max(d)$, the maximum distance in the distance matrix; $\max(f)$, the maximum flow in any of the flow matrices; $\xi(f^1, f^2)$, the correlation parameter affecting corresponding flow matrix entries of the flow matrices; fd^i, the flow dominance of the ith flow matrix; *dd*, the distance dominance; sp^i, sparsity of the ith flow matrix. Parameters (random *seed*, A, B, M, K, m and η) to the generator are also given for reference.

Chapter 3

Ant Colony Optimization

Ant Colony Optimization (ACO) is an approach to tackle $\mathcal{NP}$-hard combinatorial optimization problems. The first ACO algorithm, called Ant System (AS), was applied to the single objective QAP, yet its performance was not competitive with the state-of-the-art algorithms. Therefore, several improved ACO algorithms have been developed, e.g., Ant Colony System (ACS) and $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS). For the single objective QAP in particular, $\mathcal{MM}$AS is known to be among the best performing algorithms.

3.1 ACO Algorithms

Ant Colony Optimization (ACO) [14] is a metaheuristic inspired by the indirect communication of real ants by means of trails of a chemical substance called pheromone. Artificial *ants* are simple agents that use numerical information (artificial *pheromone information*) to communicate their experience while solving a particular problem to other ants. These principles provide a common framework for most applications of ant algorithms to combinatorial optimization problems (see [9, 10, 14] for an overview). Therefore, algorithms derived from the ACO metaheuristic are called ACO algorithms.

In ACO algorithms, an individual ant constructs *candidate solutions* to a combinatorial optimization problem by starting with an empty solution and then iteratively adding *solution components* until a complete candidate solution is generated. The ants make use of information that reflects the experience accumulated by previous ants, called *pheromone information*, and of problem dependent information, called *heuristic information*, in order to decide which solution component will be added to its current partial solution by means of a stochastic *construction policy*. Every step in an ACO algorithm at which all ants complete a feasible solution will be called *iteration*. Each ant represents unequivocally a specific solution during a certain iteration. Thus, when referring to an ant, we really mean the solution that this ant represents. After each iteration, the ants give feedback on the solutions they have constructed by depositing pheromone on solution components which they have used in their solution.[1]

The *pheromone update* strategy decides which ants are allowed to modify the pheromone information and how the selected ants modifies it. Typically, solution components which are part of better solutions or are used by many ants will receive a higher amount of pheromone and hence, will more likely be used by the ants in the following iterations of the algorithm.

Finally, in order to prevent unlimited accumulation of the pheromone trail, typically before the pheromone information is updated, all pheromone trails are decreased by a factor that models *evaporation* of the trails. This way, the pheromone trail associated with a solution component decreases exponentially if it does not receive any amount of pheromone. Evaporation enables the algorithm to "forget" bad choices over time. Additionally, some or all of the solutions constructed can be improved using local search (LS) methods.

In principle, ACO algorithms can be applied to any combinatorial optimization problem by defining solution components which the ants use to iteratively construct candidate solutions and on which they may deposit pheromone.

ACO algorithms considered in the present work follow the general algorithmic scheme outlined in Fig. 3.1. First, parameters and pheromone information are initialized. Then, a main loop is repeated until a termination condition is met, which may be a given number of solution constructions or a limit on the available computation time. In this main loop, ants construct feasible solutions by adding solutions components. Additionally, these solutions may be improved using a local search. Next, the new best solution found (*best-so-far* solution) is determined. Subsequently, a number of solutions which may include the best-so-far solution, are selected to update the pheromone information. Before updating the pheromone information, evaporation is performed by decreasing the pheromone trails by a factor ρ. Finally, when the main loop ends, the algorithm returns the best solution found since the start of the algorithm. Of course, the ACO metaheuristic is more general than this algorithmic scheme.

The first ACO algorithm was called Ant System (AS) [8, 12, 13] and was applied to the QAP. Since most of the ACO algorithms are improvements over AS, the application of AS to the QAP illustrates how other ACO algorithms can be applied to it.

[1] The pheromone information can also be updated *while* ants are constructing the solution: when each component is added to a partial solution (*online step-by-step pheromone update*) or when each solution is completed (*online delayed pheromone update*). For simplification purposes, we do not consider these alternatives.

```
SetParameters
Initialize(PhInfo)
best-so-far ← ∅
while termination condition not met do
    S ← ∅
    for each ant do
        solution ← ∅
        while solution not completed do
            component ← ConstrPolicy(PhInfo, HeurInfo)
            solution ← solution ∪ component
        end while
        S ← S ∪ {solution}
    end for
    ApplyLocalSearch( S )    # optional
    best-so-far ← FindBest( S ∪ {best-so-far} )
    B ← Select( S ∪ {best-so-far} )    # B ⊆ S ∪ {best-so-far}
    Evaporation( PhInfo, ρ )
    Update(PhInfo, B)
end while
return best-so-far
```

Figure 3.1: Algorithmic schema of ACO algorithms

3.2 Ant System Applied to the QAP

For the single objective QAP, the solution components are defined as the assignments of facilities to locations. Hence, the pheromone trail τ_{ij} corresponds to the desirability of $\phi_i = j$, that is, to assign a facility j to location i in the current solution ϕ. The order on which locations are assigned does not matter, thus when constructing a solution an ant k selects randomly or sequentially a location i not assigned yet and stochastically decides which facility j should be placed on it[2] using the following probability

$$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum\limits_{l \in \mathcal{N}_i^k} [\tau_{il}(t)]^\alpha \cdot [\eta_{il}]^\beta} \qquad \text{if } j \in \mathcal{N}_i^k \tag{3.1}$$

where $\tau_{ij}(t)$ is the pheromone information in the current iteration, η_{ij} is QAP specific heuristic information, α and β are two parameters which determine the relative importance of the pheromone and the heuristic information

[2] Alternatively, on which location should be placed the next facility. Both ways are equivalent.

respectively, and $\mathcal{N}_i^k$ is the feasible neighborhood of ant k, that is, those facilities not placed yet.

Despite the fact that specific heuristic information for the QAP can be defined [38], it is not used in most of the recent ACO approaches to the QAP [48]. This is achieved by setting $\beta = 0$ in the above equation.

This solution construction step is repeated until the feasible neighborhood is empty and, hence, a complete assignment is obtained. After all ants have completed the solution construction, the pheromone trails are updated. Evaporation is performed by first decreasing the pheromone trails by a constant factor and then ants are allowed to deposit pheromone on the assignments they have constructed. Thus, every pheromone trail is updated in the following way

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^k(t) \tag{3.2}$$

where the parameter ρ (with $0 \leqslant \rho < 1$) is the trail persistence (thus, $1-\rho$ models the evaporation) and $\Delta\tau_{ij}^k(t)$ is the amount of pheromone ant k adds to each pheromone trail. In AS, this amount is defined as

$$\Delta\tau_{ij}^k(t) = \begin{cases} 1/f(\phi^k) & \text{if } \phi_i^k = j \text{ in iteration } t \\ \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

where ϕ^k is the solution constructed by ant k and $f(\phi)$ denotes the objective value of the QAP for a solution ϕ, as defined in Eq. (2.1) on page 7.

3.3 Improvements on AS

Despite AS could be shown to perform better than other nature-inspired algorithms, such as Simulated Annealing or Genetic Algorithms, on some combinatorial problems, its performance was not competitive with the state-of-the-art algorithms [8, 13]. Therefore, many improvements over AS have been developed, such as the rank-based version of Ant System (AS$_{rank}$) [2], Ant Colony System (ACS) [11, 20] and $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) [51]. In AS$_{rank}$ only the best solution found since the start of the algorithm (best-so-far solution) and a fixed number of solutions of the current iteration are used to update the pheromone information. The better the solutions are ranked in the current iteration, the more weight they are given for the pheromone update. On the other hand, in ACS and $\mathcal{MM}$AS the best solutions found during the search are exploited by using only one solution to update the pheromones.

3.4 $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System

$\mathcal{MAX}$-$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) follows the general ACO algorithm scheme described in Fig. 3.1, but differs in three key aspects from AS.

Firstly, after each iteration only one solution is used to update the pheromone information, either the iteration-best (s^{ib}) or the best-so-far solution (s^{bf}). When using only s^{bf}, what is also proposed in ACS [11], the search may concentrate too fast around this solution and the exploration of possibly better ones is limited. On the other hand, a better exploration is achieved using s^{ib} because the iteration-best solution may differ considerably from iteration to iteration. Thus, a larger number of distinct solution components may be reinforced. This fact leads to more variability in the solutions constructed by the ants, but at the cost of a reduced exploitation of the search space around the best solution found.

Secondly, in order to prevent stagnation of the search, the range of possible pheromone trails is limited to an interval $[\tau_{min}, \tau_{max}]$. Stagnation occurs when at each iteration, one solution component has a significantly higher pheromone trail than all the other alternatives. In this case, an ant will prefer this solution component over all the others. Moreover, after the iteration, further reinforcement will be given to the solution component and evaporation will occur for all the other alternatives, aggravating the situation. $\mathcal{MM}$AS prevents stagnation by imposing explicit limits τ_{min} and τ_{max} on the minimum and maximum pheromone trails such that for all pheromone trails $\tau_{ij}(t) \in [\tau_{min}, \tau_{max}]$. Hence, the relative differences between the pheromone trails never become too extreme during the run of the algorithm. Additionally, if $\tau_{min} > 0$, the probability of choosing a specific solution component is never zero.

Lastly, the pheromone trails are initialized to τ_{max}, to achieve a higher exploration of solutions at the start of the algorithm. Actually, all the pheromone trails would be initialized to some arbitrarily high value, $\tau_{ij}(0) = \infty$, and after the first iteration will be set to $\tau_{ij}(1) = \tau_{max}(1)$, being forced to take values within the imposed pheromone limits.

Empirical results given in [51] show that $\mathcal{MM}$AS is among the best available algorithms for the QAP, where it is compared with HAS-QAP [23] (other ACO algorithm), Robust Taboo Search (RoTS) [52] and a genetic hybrid (GH) method which uses short taboo search runs for the local search [19]. $\mathcal{MM}$AS outperforms HAS-QAP for all classes of instances and RoTS and GH for structured instances and achieves a similar performance for unstructured instances (see Section 2.1.1 on page 8 for a review of the classes of instances of the QAP).

Another study described in [40] compared $\mathcal{MM}$AS with Robust Taboo Search (RoTS) [52], Reactive Taboo Search (ReTS) [1], Fast Ant Colony (FANT) incorporating local search [54, 55], Simulated Annealing (SA) [5] and a memetic algorithm described by the authors. The results showed that for unstructured instances, $\mathcal{MM}$AS achieves a similar performance as RoTS and ReTS and outperforms all the others algorithms. For structured instances, $\mathcal{MM}$AS outperformed all algorithms except the fine-tuned memetic algorithm.

Consequently, it makes sense to use the concepts of $\mathcal{MM}$AS in order to design a multi-objective ACO algorithm to tackle the bQAP.

Chapter 4

Multi-objective ACO

Despite previous research efforts on the application of ACO principles to multi-objective optimization, very few studies have actually tackled MOCO problems defined in terms of Pareto optimality. To address the design of a multi-objective ACO algorithm (MO-ACO) for this type of problem, several concepts must be reviewed. The management of the pheromone information in MOCO turns out to be a complex task that involves the definition of the pheromone information, how different pheromone information is aggregated using weights, which solutions are selected to update the pheromone information and how these solutions modify the pheromone information. In addition, some authors propose the use of multiple colonies, so that each colony weighs differently the relative importance of the multiple objectives. When multiple colonies are considered, the management of the pheromone information is even more complicated. Finally, the use of local search methods must be considered. All these features can be seen as components of a certain configuration of a general MO-ACO algorithm. With regard to the *b*QAP we consider configurations that may be related to two essentially different search strategies used to tackle MOCO problems with approximate algorithms, namely (*i*) based on dominance relations or (*ii*) based on several scalarizations of the objective vector. Finally, knowledge of the best performing ACO algorithms for single objective optimization,e.g., ACS and $\mathcal{MM}$AS, can be transferred into the multi-objective context using equivalent principles to design MO-ACO algorithms.

4.1 ACO Algorithms for MOCO Problems

Relatively few approaches of ACO for MOCO problems have been proposed so far. Moreover, many of the proposed algorithms are only applicable to problems where a lexicographic ordering of the objectives is given, i.e., where the objectives can be ordered according to their importance [22, 26, 39].

Gambardella, Taillard & Agazzi [22] studied a bi-objective vehicle routing problem with time window constraints, where the first objective, the number of vehicles, is considered to be more important than the second one, the total travel time. They proposed an algorithm that uses two ant colonies, one for each objective. A common best-so-far solution is used to update the pheromone information in both colonies. The first colony tries to find a solution with one vehicle less than the best-so-far solution while the second colony tries to improve the best-so-far solution with respect to the second objective. Whenever the first colony finds a better solution with respect to the first objective, it becomes the new best-so-far solution.

Gravel, Price & Gagné [26] tested an ACO algorithm for solving a single machine total tardiness problem with changeover cost and two additional objectives, which arises in a real-world scheduling problem for an aluminum casting center. In their approach, the heuristic information is constructed by taking all the objectives into account, however for the pheromone update only the most important objective is considered.

Mariano & Morales [39] studied a multi-colony approach for the design of water irrigation networks where for each objective there exists one colony of ants and every objective is determined knowing only the relevant part of a solution. An order is imposed on the colonies corresponding to the order of importance of the multiple objectives. In every generation, each ant from a certain colony receives a partial solution from the previous colony and tries to complete the partial solution with respect to the objective associated to that colony. When the last colony completes the solution construction, all the nondominated solutions are used to update the pheromone information.

None of the above approaches can be applied to problems where the objectives cannot be ordered according to their importance. A notable exception is the work of Iredi, Merkle & Middendorf [29], who tested a multi-colony ACO algorithm for a bi-objective single machine total tardiness problem with changeover costs. They proposed using one pheromone matrix for each objective, that is, two pheromone matrices. For the solution construction, each ant uses a weight to combine the two pheromone matrices and to combine the different heuristic information of each objective. In order to weigh the relative importance of each objective differently, each ant uses a different weight (but the same one at each iteration). Each colony has its own two pheromone matrices and set of weights and, thus, can focus the search on approximating to a certain "region" of the optimal Pareto set. Nondominated solutions found in the current iteration are used to update the pheromone matrices. Collaboration between the colonies is achieved by using the solutions of other colonies to detect dominated solutions and by using nondominated solutions from other colonies to update the pheromone matrices.

In the following sections, we examine the concepts involved in the design of an ACO algorithm for MOCO problems in terms of Pareto optimality. Many of these concepts have been considered previously in the literature. Nevertheless, we would like to formalize these concepts so that they can be applied to any MOCO problem. In addition, we discuss alternatives not yet proposed and we seek to relate these alternatives to the concepts found in the literature. All of these concepts can be seen as parameters of a certain configuration of a general multi-objective ACO (MO-ACO) algorithm. Finally, we study possible configurations of this MO-ACO algorithm in order to tackle the bi-objective QAP (bQAP).

4.2 Multi-objective Pheromone Information

Perhaps the most important step when designing an ACO algorithm for a specific problem is the definition of the pheromone and heuristic information.[1] ACO algorithms for single objective problems represent the pheromone information by a pheromone matrix (or vector) where each entry in the matrix corresponds to the desirability for a certain solution component. Thus, how the pheromone information is actually represented depends on the definition of solution components. A good example for illustrating this is the single objective QAP (cf. Section 3.2 on page 15), where a solution component is defined as the assignment of a facility j to a location i in the current solution ϕ, thus each entry τ_{ij} of the pheromone matrix corresponds to the desirability for setting ϕ_i to j ($\phi_i = j$).

In contrast, each objective in a MOCO problem may define the solution components differently. For instance, let us consider a bi-objective scheduling problem, where the first objective depends on which position of the schedule a certain job is placed, whereas the second objective depends on the relative position of a certain job with respect to the previous jobs.[2] In this case, solution components for the first objective can best be defined as assignments of jobs to positions in the schedule, whereas for the second objective they are defined depending on the predecessor/successor relationship among jobs. Therefore, given a feasible solution ϕ to this problem, which represents a certain sequence of jobs, the pheromone information for the first objective is the desirability that job j is on place i of the schedule ($\phi_i = j$), whereas the pheromone information for the second objective corresponds to the desirability that job j comes immediately after job h in the schedule ($\phi_i = j \wedge \phi_{i-1} = h$). The pheromone information for both objectives cannot be expressed as one pheromone matrix because an entry τ_{ij} of the pheromone matrix cannot represent the desirability for both solution components at the same time. Moreover, if the total number of jobs is larger than the number of jobs that can be scheduled at one time, then the dimension of the pheromone matrix for the first objective is different from the one for the second objective.

On the other hand, for the multi-objective QAP (mQAP), the pheromone information for all the objectives corresponds to the desirability that a facility j is assigned to location i in the current solution ϕ ($\phi_i = j$), that is,

[1] In the following sections we mainly study the definition of the pheromone information. Yet, the conclusions can also be applied to the heuristic information.

[2] This is the case, for example, for the Single Machine Total Tardiness Problem with Changeover Costs that is described in [29].

all objectives define the solution components in a unique manner. Hence, in other multi-objective optimization problems, solution components and, consequently, pheromone information may have a single meaning.

In general, in the case of MOCO problems, there may be as many different meanings for the pheromone information as the number of objectives.

In order to represent the pheromone information for a MOCO problem, an initial approach may be to keep the pheromone information for only one of the objectives. Then, solution components are defined only for this objective and the pheromone information is represented consequently as if this objective was the only one considered in the problem. This is the approach used by some ACO algorithms [22, 26] for MOCO problems where one objective is considered more important than the others. As would be expected, when only one pheromone matrix is considered for a certain objective, solutions obtained are better for that objective to the detriment of the remaining objectives [29]. Thus, this approach is not adequate without *a priori* knowledge about the relative importance of the different objectives.

For this reason, we will discuss two different approaches to defining the pheromone information when objectives cannot be ordered by importance. The first one, proposed by Iredi et al. [29], uses multiple pheromone information, such that the pheromone information is defined differently for each objective, whereas in the second approach, we propose the definition of a single pheromone information for all the objectives simultaneously.

4.2.1 Multiple Pheromone Information

We will assume that a MO-ACO algorithm uses *multiple pheromone information* when different pheromone information is defined for each objective and *weights* are used to aggregate them into a single value, in a similar way that weights are used to aggregate different objectives in the weighted sum scalarization of a multi-objective problem (cf. Section 1.4 on page 4).

Since each objective may differently define the solution components, different values may be needed by an ant in order to define the possible solution components that can be added to its partial solution. These values can be considered as components of a *state vector* which defines the current state of the partial solution being constructed by an ant. In this state vector S, two objectives (q and r) with different definitions of the solution components will have different values on their corresponding elements of the state vector ($S_q \neq S_r$), whereas if they have the same definition of a solution component then they will have the same value ($S_q = S_r$). As well, an objective q may define the solution components only in terms of the next element j to be added to the partial solution. Consequently, their corresponding element of the state vector will be null ($S_q = \varnothing$).

Let us assume an arbitrary MOCO problem with Q objectives. Then, an ant k adds j to its partial solution with the following probability:

$$p_{Sj}^{k} = \frac{\left[\prod\limits_{q=1}^{Q}\left(\tau_{S_qj}^{q}\right)^{\lambda_q}\right]^{\alpha}\cdot\left[\prod\limits_{q=1}^{Q}\left(\eta_{S_qj}^{q}\right)^{\lambda_q}\right]^{\beta}}{\sum\limits_{l\in\mathcal{N}_S^k}\left(\left[\prod\limits_{q=1}^{Q}\left(\tau_{S_ql}^{q}\right)^{\lambda_q}\right]^{\alpha}\cdot\left[\prod\limits_{q=1}^{Q}\left(\eta_{S_ql}^{q}\right)^{\lambda_q}\right]^{\beta}\right)} \quad \text{if } j\in\mathcal{N}_S^k \tag{4.1}$$

where $S = \{S_1, \ldots, S_Q\}$ is the state vector of the current partial solution, and each element S_q of this vector is value needed by each objective $q = 1, \ldots, Q$ to define a solution component; $\mathcal{N}_S^k$ is the feasible neighborhood of ant k given the current state vector S; $\tau_{s_qj}^{q}$ is the pheromone information of j given S_q for the q^{th} objective in the current iteration and, in an equivalent way, $\eta_{S_qj}^{q}$ is the heuristic information for j given S_q that corresponds to the q^{th} objective. Finally, λ_q is a value that weighs the relative importance of the q^{th} objective and it can be seen as the q^{th} component of a weight vector λ taken from the set Λ, as defined in (1.2) on page 4. Thus, when $\lambda_q = 0$ the q^{th} objective is not considered and when $\lambda_q = 1$ it is the only one considered.

For instance, in the bi-objective scheduling problem described above, each solution component for the first objective is an assignment of a job to a position, whereas for the second objective it is an assignment of a job to the previous job scheduled. Thus, in order to decide which the next job j scheduled is, an ant needs to know at which position i the job should be scheduled and which the previous job h is. Therefore, in this case, the current state of a partial solution is $S = \{i, h\}$. Following Eq. (4.1) with $Q = 2$, an ant k adds a job j to its current schedule with probability:

$$p_{\{i,h\}j}^{k} = \frac{\left[\left(\tau_{ij}^{1}\right)^{(1-\lambda)}\cdot\left(\tau_{hj}^{2}\right)^{\lambda}\right]^{\alpha}\cdot\left[\left(\eta_{ij}^{1}\right)^{(1-\lambda)}\cdot\left(\eta_{hj}^{2}\right)^{\lambda}\right]^{\beta}}{\sum\limits_{l\in\mathcal{N}_{\{i,h\}}^k}\left(\left[\left(\tau_{il}^{1}\right)^{(1-\lambda)}\cdot\left(\tau_{hl}^{2}\right)^{\lambda}\right]^{\alpha}\cdot\left[\left(\eta_{il}^{1}\right)^{(1-\lambda)}\cdot\left(\eta_{hl}^{2}\right)^{\lambda}\right]^{\beta}\right)} \quad \text{if } j\in\mathcal{N}_{\{i,h\}}^k$$

where $\mathcal{N}_{\{i,h\}}^k$ is the feasible neighborhood of ant k, that is, those jobs that have not been already scheduled; τ_{ij}^1 and η_{ij}^1 are the pheromone and specific heuristic information for the first objective; τ_{hj}^2 and η_{hj}^2 are the pheromone and specific heuristic information for the second objective; and finally, $\lambda \in [0, 1]$ is a value that allows the weighing of the relative importance of the two objectives differently.

In the case of the mQAP, the solution components for all objectives are defined as assignments of facilities to locations. Thus, in order to decide the next facility j to place, an ant only needs to know at which location i the facility is going to be placed, therefore $S = \{i\}$. For simplicity, let us restrict ourselves to the bi-objective case ($Q = 2$). Following Eq. (4.1), an ant k

places facility j with probability:

$$p_{ij}^{k} = \frac{\left[\left(\tau_{ij}^{1}\right)^{(1-\lambda)} \cdot \left(\tau_{ij}^{2}\right)^{\lambda}\right]^{\alpha} \cdot \left[\left(\eta_{ij}^{1}\right)^{(1-\lambda)} \cdot \left(\eta_{ij}^{2}\right)^{\lambda}\right]^{\beta}}{\sum\limits_{l \in \mathcal{N}_i^k} \left(\left[\left(\tau_{il}^{1}\right)^{(1-\lambda)} \cdot \left(\tau_{il}^{2}\right)^{\lambda}\right]^{\alpha} \cdot \left[\left(\eta_{il}^{1}\right)^{(1-\lambda)} \cdot \left(\eta_{il}^{2}\right)^{\lambda}\right]^{\beta}\right)} \quad \text{if } j \in \mathcal{N}_i^k \tag{4.2}$$

where $\mathcal{N}_i^k$ is the feasible neighborhood of ant k, that is, those locations which are still free; τ_{ij}^1 and τ_{ij}^2 are the pheromone information in the current iteration for each of the two objectives of the bQAP; η_{ij}^1 and η_{ij}^2 are QAP specific heuristic information for each of the two objectives; and finally, as before, $\lambda \in [0, 1]$ is a value that allows us to weigh the relative importance of the two objectives differently.

Equation (4.2) illustrates a potential issue when using multiple pheromone information for problems where all the objectives define the solution components in the same manner. In this type of problem, when updating the multiple pheromone matrices, if the solution components receive the same amount of pheromone independently of the particular objective then it occurs that $\tau_{ij}^1 = \tau_{ij}^2 = \tau$, and thus the aggregation of the pheromone matrices is useless since $\tau^{(1-\lambda)} \cdot \tau^{\lambda} = \tau$. Therefore, when multiple pheromone items are defined in terms of equivalent solution components, the pheromone update strategy must ensure that the multiple pheromone items are in fact different. We will address this issue when discussing the pheromone update strategies (Section 4.3 on page 27).

Finally, an important issue is the definition of the weight vector. Multiple pheromone information is aggregated into a single value using *weights*, or more formally a weight vector λ as in Eq. (4.1). This weight vector regulates the relative importance of the different objectives. We are not interested in obtaining a single solution but a Pareto set. Thus, we cannot use only one weight vector but a finite set $\Gamma \subset \Lambda$ of *maximally dispersed* weight vectors [47], i.e., the weight vectors in Γ are as much distributed as possible in the infinite space defined by Λ.

Iredi et al. [29] proposed assigning each weight vector $\lambda_k \in \Gamma$ to a different ant k. Thus, the whole set of weight vectors is used at every iteration. As well, in the ACO algorithms we are considering (AS and $\mathcal{MM}$AS), at any given iteration each ant constructs its solution independently from the others. Therefore, it does not matter the order in which weights are assigned to ants. An alternative method would be to assign each weight vector $\lambda_t \in \Gamma$ to a different iteration t. In this case, all the ants use the same weight at a given iteration and a different weight in the following iteration. Since ants from a given iteration are not independent from those in the previous iteration, it makes sense in this case to assign each weight to each iteration in such an order that two consecutive weight vectors differ only by $\pm z$ in any two components [47], where z is the minimum difference between any two values of any component for all the weight vectors in Γ.

The weight vector can be seen as a search "direction" in the objective space. Then, the first method searches in *all possible directions* at each iteration, while the second method tries to find a good solution in *one direction* at one iteration and using this solution as a starting point in the next iteration but in a slightly different direction. The best performing method will depend on the underlying multi-objective landscape [31].

4.2.2 Single Pheromone Information

We will assume that a MO-ACO algorithm uses *single pheromone information* when the pheromone information is defined in a unique manner for all objectives and thus weights are not needed. The problem once again is how to define the pheromone information for those objectives where the solution components have a different meaning. The approach to solve this problem will be similar to using multiple pheromone information, that is, to use a state vector S that defines the current state of the partial solution being constructed by an ant.

Therefore, when using single pheromone information, the pheromone information of a solution component is expressed as τ_S^j for all the objectives, which is the desirability for j given the state S. Then, an ant k adds j to its partial solution with the following probability:

$$p_{S,j}^k = \frac{\left[\tau_{Sj}\right]^\alpha \cdot \left[\eta_{Sj}\right]^\beta}{\sum\limits_{l \in \mathcal{N}_S^k} \left(\left[\tau_{Sl}\right]^\alpha \cdot \left[\eta_{Sl}\right]^\beta\right)} \qquad \text{if } j \in \mathcal{N}_S^k \tag{4.3}$$

where $[\tau_{Sj}]$ is a multi-dimensional matrix with as many dimensions as elements of the state vector S, namely the number of ways that solution components may be defined by the objectives, plus one more dimension for j.

For the bi-objective scheduling problem, the state vector contain the position i where a new job will be scheduled and also the job h that was scheduled in the previous position, $S = \{i, h\}$. Therefore, the single pheromone information is represented by a pheromone matrix $[\tau_{ihj}]$ with dimensions $I \times J \times J$ where I is the size of the schedule and J is the number of jobs. Then, applying Eq. (4.3) with $Q = 2$, an ant k adds a job j to its current schedule with probability:

$$p_{ihj}^k = \frac{\left[\tau_{ihj}\right]^\alpha \cdot \left[\eta_{ihj}\right]^\beta}{\sum\limits_{l \in \mathcal{N}_{ih}^k} \left(\left[\tau_{ihl}\right]^\alpha \cdot \left[\eta_{ihl}\right]^\beta\right)} \qquad \text{if } j \in \mathcal{N}_{ih}^k$$

In the case of the mQAP, this approach is even more natural, since a solution component has a unique meaning for all objectives and the state vector has only one element, i.e., the location i where the next facility should be placed, $S = \{i\}$. Therefore, the single pheromone information is represented by a pheromone matrix $[\tau_{ij}]$ as in AS and $\mathcal{MM}$AS for the single objective QAP.

4.2.3 Computational Efficiency Aspects of Multi-objective Pheromone Information

In terms of computational resources required, at first it might seem that using single pheromone information is very expensive because of the use of a multidimensional matrix, but there are some situations in which the use of single pheromone information is more efficient in terms of both space and computation time required than the use of multiple pheromone information.

From the point of view of required space, single pheromone information is represented with a multi-dimensional matrix with as many dimensions as the number of manners that solution components may be defined by the Q objectives, plus one more dimension. This number is equivalent to the size of the state vector $1 \leq |S| \leq Q$ plus one. Therefore, the single pheromone information leads to a memory requirement of $\mathcal{O}\big(n^{|S|+1}\big)$, where n is the largest dimension of the pheromone matrix. In the worst case all the Q objectives define the solution components differently and then, the single pheromone information is represented with a Q-dimensional matrix, leading to a memory requirement of $\mathcal{O}\big(n^{Q+1}\big)$. On the other hand, multiple pheromone information is represented in any case with Q different matrices, leading to a memory requirement of $\mathcal{O}\big(Q \cdot (n \cdot n)\big)$. However in some problems, as we have seen for the mQAP, the number of meanings of the solution components may be small compared to the number of objectives, and therefore the memory requirements for using multiple pheromone information are much higher than for using single pheromone information.

With regard to computation time, when multiple pheromone information is used, the calculation of p^k_{Sj} implies the evaluation of $\prod_{q=1}^{Q} \big(\tau^q_{S_q j}\big)^{\lambda_q}$. Consequently, the computation time required is $\mathcal{O}(Q)$. Moreover, the power and the product are very expensive operations in terms of computation time. Hence, there would be a large multiplicative constant hidden in the previous expression. On the other hand, when single pheromone information is considered, there is only one pheromone value, thus the evaluation takes a constant time without expensive operations. As well, the evaporation procedure implies that all the entries of the pheromone matrices must be modified. Therefore, when single pheromone information is used, the computation time of the evaporation procedure is $\mathcal{O}\big(n^{|S|+1}\big)$ where n is again the largest dimension of the pheromone matrix, whereas for multiple pheromone information the computation time is $\mathcal{O}\big(Q \cdot (n \cdot n)\big)$.

In conclusion, which definition of the multi-objective pheromone information actually requires fewer computational resources depends on the number of objectives of the particular problem being considered and how these objectives define the solution components.

4.3 Pheromone Update Strategies

The best performing ACO algorithms for single objective problems typically update the pheromone information by using only the best or a few of the best solutions found in the current iteration (*iteration-best* strategy) or since the start of the algorithm (*best-so-far* strategy) [11, 51]. In multi-objective problems, the best solutions can also be taken from a candidate set including all solutions found in the current iteration $\mathcal{C}^{\text{ib}}$ or since the start of the algorithm $\mathcal{C}^{\text{bf}}$. The real difficulty lies in the definition of the best solutions of the candidate set.

With a single objective, the comparison criterion between solutions is evident because the best solution is that one with the best value of the objective function. On the other hand, with multiple objectives, the straightforward criterion is the Pareto optimality and thus the best solutions of the candidate set (either $\mathcal{C}^{\text{ib}}$ or $\mathcal{C}^{\text{bf}}$) are those that are nondominated, or in other words, those solutions that belong to the Pareto set. We will refer to this strategy as *selection by dominance*.

When using selection by dominance and multiple pheromone information, if each objective defines the solution components in a different manner, the selected solutions update each pheromone matrix differently because each solution component has a different meaning for every pheromone matrix.[3] However, a problem arises when different pheromone matrices define the solution components in the same manner because they will be reinforced equally, becoming the same matrix, causing the aggregation to turn out to be useless. Furthermore, when using selection by dominance and multiple pheromone information, the meaning of each pheromone matrix is no longer the desirability of a certain objective, but for a certain set of solution components (which are defined by each pheromone information) and then the weight vector regulates the relative importance of different solution components.

On the other hand, when selection by dominance is used with single pheromone information, the pheromone trails of the solution components that belong to nondominated solutions are reinforced in a natural way. There is exactly one entry of the single pheromone information for each possible solution component, thus each selected solution will deposit pheromone in the entries of the single pheromone information associated with its solution components.

Alternatively, we can define a weight vector and select the best solution with respect to a weighted sum scalarization of the multiple objectives. However, this *selection by scalarization* is even more problematic because *(i)* for

[3] This is exactly the strategy followed by the algorithm proposed by Iredi, Merkle & Middendorf [29].

each weight vector all the solutions are scalarized and compared in order to select the best solution with respect to the weight vector *(ii)* it is not clear how the selected solutions should update the multi-objective pheromone information, particularly when multiple pheromone information is considered. Therefore, we will restrict ourselves to a simplified form of the selection by scalarization. In this simpler strategy, only the best solutions with respect to each objective are selected to update the pheromone information. Then, when multiple pheromone information is considered, each pheromone matrix associated with each objective is updated by the solution with the best objective value for the respective objective. Consequently, this simplified form of the selection by scalarization is called *selection by objective*.

Selection by objective has two main advantages over selection by scalarization. First, only one ant per pheromone matrix will be allowed to deposit pheromone as in $\mathcal{MM}$AS and ACS, which has led to improved performance over the original AS (see Section 3.3 on page 16). As a result, advanced techniques used in these algorithms can be easily adapted to multi-objective problems. Second, each pheromone matrix focuses on one objective, thus the aggregation of all of them by means of a weight vector truly regulates the relative importance of each objective. This is not the situation in the selection by dominance when using multiple pheromone information. Therefore, the problem described above which occurs when the solution components have the same meaning for different pheromone information is avoided with this strategy, because each pheromone matrix is updated by a different solution and thus the pheromone matrices are actually different.[4] In this way, the concept of multiple pheromone information can be applied to MOCO problems similar to the mQAP.

It must be noted that when two solutions both have the best value for a certain objective, the strategy of selection by objective always chooses the one that dominates the other. However, with $Q \geqslant 3$ it may happen that two nondominated solutions have the best value for an objective. In this situation, either we can choose one of them or we can select both, although in the latter case more than one ant is allowed to deposit pheromone and we give up one of the properties of this strategy.

Finally, the amount of deposited pheromone must be defined. The only restriction is that the solution cost cannot be used at all because the values of different objective functions are not comparable. When multiple pheromone information is considered, we must take special care to ensure that the amount of deposited pheromone is independent from the pheromone matrix on which it is deposited, otherwise some objectives are implicitly consid-

[4] Of course, it may happen that an ant is the best of the candidate set with respect to more than one objective and then it will deposit pheromone in more than one pheromone matrix. When this situation occurs frequently, we should ask ourselves if using multiple pheromone information is the correct approach to solve that specific problem.

ered more important than others. Taking these restrictions into account, the amount of deposited pheromone can be defined in any manner. Finally, with regard to the pheromone evaporation procedure, the only difference between the multi-objective and the single objective case is the size and the number of pheromone matrices, thus in principle any method can be used although efficient techniques would of course be preferred.

4.4 Multiple Colonies

The image of the optimal Pareto set in the objective space is a trade-off surface between the different objectives. Over this surface, two solutions can be said to belong to different *regions* with respect to the differences in their objective vectors, e.g., if the distance between their respective objective vectors is more than a given value or if we divide the objective space somehow into hyper-boxes such that their objective vectors belong to different hyper-boxes. Notice that the solution vector in the search space is not considered, thus the desirable solution components of two Pareto optimal solutions belonging to different regions could be very different or not. Nevertheless, as the real correlation between the objectives decreases, it is to be expected that solution components that are desirable for minimizing one objective will not be desirable for minimizing the others. Therefore, it could be advantageous to focus the search on a certain region, independently from the search in other regions.

In a multi-colony approach, the total number of ants is divided into disjoint sets, each of these sets being called a colony. In the following, we will assume that every colony has the same number of ants, or more formally, that there are m ants, c colonies and $\frac{m}{c}$ ants per colony. Multi-colony ant algorithms have been previously used to parallelize ACO algorithms in order to solve single objective problems. In the multi-objective context, the main idea behind the multi-colony approach is not parallelization, but the fact that different colonies can weigh the objectives differently [29]. In other words, each colony can be seen as trying to approximate a different region of the optimal Pareto set.

Each colony is independent from the others in the sense that it has its own ants and pheromone information, such that each ant from a certain colony constructs its solution guided only by the pheromone information from its own colony. If no cooperation takes place between colonies, then we would have a multi-start single colony ant algorithm where at each restart the algorithm tries to approximate a different region of the optimal Pareto set. Nonetheless, the solutions found by one colony can help to identify *weak* solutions (i.e., dominated solutions) found by other colonies and one colony can find good solutions by chance which belong to the region of other colonies. Therefore, cooperation between colonies can increase the overall performance of the algorithm. Cooperation is achieved by exchanging solutions between colonies so the updating of the pheromones of one colony is influenced by solutions from other colonies.

Three aspects define the behavior of the multiple colonies: *(i)* the set of weight vectors used to aggregate multiple pheromone information can force each colony explicitly to approximate a different region of the optimal Pareto set, *(ii)* a candidate set formed by solutions from all colonies enforces cooperation between colonies, and *(iii)* the pheromone update strategy must select the colony where each solution updates the pheromone information.

4.4.1 Weight Vectors in the Multi-colony Approach

The use of single pheromone information does not in itself force each colony to focus on a certain region. When multiple pheromone information is used, on the other hand, the set of weight vectors that each colony uses in order to aggregate its multiple pheromone information defines in some way a region in the objective space on which the colony focuses the search.

The infinite set Λ defines all the possible directions that can be taken to approximate the optimal Pareto set. Any finite subset Γ of *maximally dispersed* weight vectors defines a region for the whole optimal Pareto set.[5] Thus, a partition of Γ defines regions in the optimal Pareto set that can be either disjoint or overlapping regions depending on whether disjoint partitions of Γ are considered or not. Then, the multiple colonies can use the same set Γ, disjoint or overlapping partitions of it.

For the bi-objective case, a single value λ is enough to define each weight vector $\{1-\lambda, \lambda\}$. Then, given the number of colonies c and the number of weight vectors per colony w, Iredi et al. [29] proposed the following possibilities to define each weight value λ_{ki} with $k = 1, \dots, w$ and $i = 1, \dots, c$

– *Single Region:* for all colonies the values of λ are in the interval $[0, 1]$.

$$\lambda_{ki} = \frac{k-1}{w-1}$$

– *Disjoint Regions:*

$$\lambda_{ki} = \frac{(i-1)w + (k-1)}{c \cdot w - 1}$$

– *50% Overlapping regions:* the interval of values of λ for colony i overlaps by 50% with the interval for colony $i-1$ and colony $i+1$.

$$\lambda_{ki} = \frac{(i-1)(w-1) + 2(k-1)}{(w-1)(c+1)} \tag{4.4}$$

[5] Nonsupported solutions cannot be obtained using a weighted sum scalarization. Therefore, a region that only contains nonsupported solutions cannot be defined using weight vectors.

4.4.2 Candidate Set in the Multi-colony Approach

Turning to the candidate set from which the best solutions are selected in order to update the pheromone information, there are two alternatives: either the set is formed with solutions from all colonies or each colony defines its own set with its own solutions. When the candidate set is formed by solutions of only one colony at a time, there is no cooperation mechanism because the decision about which solutions are selected to update the pheromone information only depends on the solutions found in that colony. Nevertheless, cooperation can still be imposed by other mechanisms, simply by exchanging the selected solutions between colonies or even more complicated methods that have been studied for parallelized ACO algorithms. In the present work, restrictions on communications between colonies are not considered.[6] Hence, the cooperation between colonies is enforced by using a candidate set formed with solutions from all colonies.

As for the single colony approach, the candidate set can be the set of all solutions obtained by all colonies in the current iteration $\mathcal{C}^{\text{ib}}$ (iteration-best strategy) or since the start of the algorithm $\mathcal{C}^{\text{bf}}$ (best-so-far strategy).

4.4.3 Pheromone Update Strategies with Multiple Colonies

The pheromone update strategies described for the single colony approach can also be used with multiple colonies. To enforce the specialization of the colonies, each ant deposits pheromone on only one colony.

The *selection by dominance* method is straightforwardly adapted to the multi-colony approach, i.e., the ants belonging to the Pareto set of the candidate set are distributed between colonies and are allowed to deposit pheromone.

The case of the *selection by scalarization* is somewhat more complicated. Let us consider only the simplified method described above, namely *selection by objective.* In the single colony approach, this method updates the pheromone information using the best solution with respect to each objective. Solution components of these selected solutions will have a high probability of belonging to solution constructed in the next iteration. Since all of the selected solutions update the same pheromone information, solution components which belong to more than one selected solution at one time will have a higher probability of being part of solutions constructed in the next iteration. However, if the selected solutions are distributed between several colonies, the solution components common to solutions which are the best with respect to the different objectives will not have a higher probability. Furthermore, when multiple pheromone information and multiple colonies are considered, some pheromone matrices are not updated. In this case, for each colony there is

[6] Restrictions on cooperation can come from distributed systems where communications may be expensive.

one pheromone matrix for each objective. Then, when using selection by objective, if solutions selected from the candidate set are distributed among colonies, the number of selected solution will be much smaller than the number of pheromone matrices which must be updated because the maximum number of solutions selected to update is equal to the number of objectives.

For these reasons, we propose a definition of selection by objective with multiple colonies which distributes the candidate solutions among colonies before selecting the best solutions.[7] Under this definition, the minimum number of ants allowed to update the pheromone information in a colony is equal to the number of objectives. As well, since an ant cannot update the pheromone information of more than one colony, the selection by objective method enforces the specialization of the colonies.

The selection by scalarization with a single colony also assures that a selected solution always belongs to the Pareto set of the candidate set. As well, the collaboration between colonies in the multi-colony approach aims to detect *weak* solutions (dominated by known solutions). In order to also have these properties when selection by scalarization is used with multiple colonies, the set that is distributed among colonies is actually the Pareto set of the solutions from all the colonies.

In summary, in *selection by objective*, the Pareto set of the candidate set is somehow distributed among the colonies and then for each colony the best solution with respect to each objective is allowed to deposit pheromone either on the single pheromone matrix (when single pheromone information is considered) or on the pheromone matrix associated with the respective objective (when multiple pheromone information is considered). Finally, when the number of colonies is one, this definition of selection by scalarization is consistent with the one previously proposed.

Whenever multiple colonies are considered, the pheromone update strategy must also decide on which colony each solution updates the pheromone information, i.e., it must define how solutions are distributed among the multiple colonies. Iredi *et al.* [29] proposed two methods. The first strategy, called *update by origin*, allows each ant to deposit pheromone only in the colony from which that ant comes. Hence, this is not essentially a collaborative strategy because it does not enforce the exchange of solutions between colonies. Moreover, it does not explicitly force the colonies to focus on different regions. The second method is called *update by region* and explicitly forces each colony to focus on a different region. The regions are defined implicitly by sorting the Pareto set of the candidate set and then dividing it as equally as possible into parts that are distributed among the colonies. This method can only be used with bi-objective problems because for more than

[7] The drawback of this approach is that if the number of solutions assigned to a colony is less than the number of objectives, then a solution will update the pheromone information of the colony more than one time, however this can also occur with a single colony.

two objectives a set of non-dominated objective vectors can only be *partially* sorted. More formally, let $\mathcal{C}^*$ be the Pareto set of the candidate set (either $\mathcal{C}^{\text{ib}}$ or $\mathcal{C}^{\text{bf}}$) and let c be the number of colonies. Firstly, the *update by region* strategy sorts $\mathcal{C}^*$ in order of increasing values of the first objective. Next, $\mathcal{C}^*$ is divided into disjoint partitions $\mathcal{C}^*_1, \mathcal{C}^*_2, \ldots, \mathcal{C}^*_c$ in such a way that their respective sizes differ by at most one.[8] Lastly, all solutions in $\mathcal{C}^*_i$ update the pheromone information of colony i.

The order in which the solutions are sorted only matters if we are using another mechanism to force colonies to focus on different regions. For instance, when using multiple pheromone information, the weights can be defined to force colonies to focus on different regions. Therefore, if the ants from the first colony construct solutions considering more importance to the first objective because of the weight vectors and the ants from the colony c attach more importance to the second objective, then the solutions in $\mathcal{C}^*$ must be sorted in increasing order of the first objective.

4.5 Local Search Methods for MO-ACO

For many single objective combinatorial optimization problems, ACO algorithms obtain the best performance only when the solutions constructed by the ants are improved using local search methods [11, 21, 38, 48, 49, 50]. Furthermore, the best configuration of settings and parameters for ACO algorithms depends greatly on whether local search is used or not. Therefore, when local search can be efficiently applied to a certain problem and it may improve the performance of an ACO algorithm, then the algorithm should be analyzed, taking into account the use of local search since the beginning of the analysis.

Local search methods for MOCO can be based on methods for single objective problems applied to several scalarizations of the objective vector of the MOCO problem. On the other hand, they can use the Pareto optimality criteria to keep an archive of Pareto locally optimal solutions, as done by Pareto Local Search (PLS) [42, 43]. In the former case, the number of Pareto locally optimal solutions obtained is bounded by the number of scalarizations considered. In the latter case, the size of the optimal locally Pareto set may be exponential in the instance size. For this reason, we propose adapting the concepts for bounded archiving in multi-objective evolutionary algorithms (MOEA) [33, 37] in order to obtain a more efficient variant of PLS.

[8] Notice that $|\mathcal{C}^*| = c \cdot p + r$, hence each partition $\mathcal{C}^*_1, \ldots, \mathcal{C}^*_r$ has size $p + 1$ and each partition in $\mathcal{C}^*_{r+1}, \ldots, \mathcal{C}^*_c$ has size p.

4.5.1 Local Search for Single Objective Problems

Local search methods for single objective problems can be applied to their multi-objective variants because any MOCO problem can be transformed into a single objective problem using a weighted sum scalarization of the objective vector. Therefore, local search methods for MOCO problems may be based on several scalarizations of the objective vector by altering the weight vectors.

It is well-known that globally optimal solutions for scalarized problems are also Pareto globally optimal solutions. Furthermore, local optima for scalarized problems where all the weights are positive are also local Pareto optima,i.e., there is no other solution in its neighborhood that dominates it [42]. However, these local search methods will not be able to find nonsupported solutions.

4.5.2 Pareto Local Search (PLS)

Local search methods for MOCO problems can be based on the Pareto optimality criterion. For example, Pareto Local Search (PLS) [43] is a multi-objective extension of local search algorithms for single objective problems which uses the Pareto optimality as the acceptance criterion. PLS starts from a solution and examines its neighborhood. Next, any nondominated solution found is added to an archive and the dominated ones are removed from it. PLS terminates when all the neighboring solutions of all solutions in the archive have been explored. The size of the locally optimal Pareto set obtained may be exponential in the instance size. Thus, for some problems or specifically in some instances, because of the large size of the archive, PLS may be very inefficient in terms of space and computation time required, even more so if it is to be used by a MO-ACO algorithm.

4.5.3 Bounded Pareto Local Search (BPLS$_A$)

In order to address the previously described problem, we propose a *bounded* variant of Pareto Local Search (BPLS$_A$), which is a simplified approach of concepts previously studied in MOEA [33, 37].

The only difference with respect to PLS is the procedure which adds a new neighborhood solution to the archive. This procedure ensures an upper bound to the size of the archive by placing a hyper-grid in the objective space and only allowing one objective vector to be in each grid cell at one same time [37]. When a new solution is added to the archive, there are three possibilities: *(i)* the solution is dominated by any in the archive; it is discarded and thus the size of the archive does not change; *(ii)* the solution dominates at least one solution in the archive; all dominated solutions are discarded and the new solution is added to the archive; thus the size of the archive does not change or may even decrease; or *(iii)* in any other case, the solution is compared with the ones in the archive in order to ensure that there is no other solution in the same cell.

Each cell is represented by a *box vector* and two solutions belong to the same cell when they have the same box vector. Given A as the upper bound of the size of the archive, then the number of divisions at each axis is given by $\frac{A+1}{2}$. Thus, for any solution with objective vector $a = \{a_1, \ldots, a_Q\}$, its corresponding box vector is $B(a) = \{B_1, \ldots, B_Q\}$, with

$$B_i = \left\lfloor \frac{(a_i - min_i) \cdot (A+1)}{2(max_i - min_i)} \right\rfloor \quad i = 1, \ldots, Q$$

where max_i and min_i are the respective maximum and minimum values of objective i for all solutions in the archive and $\frac{A+1}{2(max_i - min_i)}$ is the length of each grid cell in the dimension i.

The ranges (max_i, min_i) are adapted over time [36]. When a new solution with objective vector $a = \{a_1, \ldots, a_Q\}$ is added to the archive and for any objective i a value is lower than the respective minimum range $(a_i < min_i)$, then the respective minimum range is updated $(min_i = a_i)$. In this case, the maximum values are also checked and updated, if any value of the remaining objectives is greater than its respective maximum value $(a_j > max_j$, $j = 1, \ldots, Q \wedge j \neq i)$. Whenever the ranges for objective i change, the corresponding components of the box vector for every solution in the archive must be calculated again and all the box vectors must be compared to ensure that every solution belongs to a different grid cell.

In the case where two solutions belong to the same grid cell, one of them should be removed. In order to keep the "tails" of the archive, solutions with an objective value equal to any minimum are protected from removal. In any other case, we keep the oldest solution.

4.6 MO-ACO Applied to the *b*QAP

An ACO algorithm applied to a certain MOCO problem may use any combination of the concepts described above. A certain combination of concepts or strategies can be seen as a particular *configuration* of a general MO-ACO algorithm. Therefore, we would like to test several configurations in order to find the best one. Ideally, the available knowledge from previous or similar experiments would help us to decide which configurations may be interesting to consider. Nowadays, there is very limited knowledge about the mQAP or about MO-ACO algorithms. In contrast, there is a vast literature on multi-objective evolutionary algorithms (MOEA). Two essentially different search strategies are usually considered in MOEA, either based on the dominance criteria (class D) or based on several scalarizations of the objective vector (class S). Consequently, in the study of MO-ACO applied to the bQAP, we will focus on straightforward configurations which reflect these different types of search strategies.

The use of multiple pheromone matrices which are aggregated by means of weights resembles the scalarization of the multiple objective functions. On the other hand, when single pheromone information is considered, there is no need of weights. Moreover, if we use the dominance criteria to select the best solutions that will update the single pheromone information (selection by dominance), then the search strategy is based only on the dominance criteria. For the bQAP in particular, the selection by dominance method does not suggest the utilization of multiple pheromone matrices in a natural manner simply because a solution component has a unique meaning. Thus, some other mechanism is needed to distinguish between the multiple pheromone matrices. In contrast, the strategy of selection by objective allows us to use multiple pheromone information. For these reasons, we consider a configuration of class D that comprises the use of single pheromone information and selection by dominance, and a configuration of class S that comprises the use of multiple pheromone information (Eq. 4.2) and selection by objective. In addition, weights can be defined so that the algorithm searches in all directions or in one direction at each iteration. Therefore, we consider search configurations of class S_{all} and S_{one}, respectively.

The available knowledge about ACO for the single objective QAP indicates that multi-objective local search methods may improve the performance of the MO-ACO algorithm applied to the bQAP. Therefore, we will study configurations which apply local search to the solutions constructed by the ants versus configurations which do not use local search. In order to be consistent with the search configuration of class D and class S, we use two essentially different local searches for each class but with the following similarities. Firstly, both local searches use the same underlying 2-exchange neighborhood (*2-opt*), that is, the set of all solutions where the location of two different facilities is exchanged. Secondly, both explore the whole neighborhood before applying the acceptance criterion, that is, they are best improvement local searches. Lastly, both make use of the fast delta evaluation of the bQAP, which is well-known in the single objective case [52] and was extended to the mQAP by Paquete & Stützle [43]. In contrast, the local search method for class S is based on the weighted sum scalarization of the objective function vector (W-LS), whereas for class D is based on Pareto Local Search (PLS) or its bounded variant (BPLS_A). For W-LS, in particular, each solution is improved by W-LS using the same weight that the ant used before while constructing the solution. The various classes of search strategy are summarized in Table 4.1.

We would also like to compare the multi-colony approach to the use of a single colony. Hence, we consider a cooperative multi-colony approach where the exchange of solutions between colonies is performed using the "update by region" strategy and the candidate set is comprised of solutions from all colonies. In particular, when combined with search configurations of class S, the set of weight vectors is defined using 50% overlapping regions (Eq. 4.4). A similar configuration has already been tested over a bi-objective scheduling problem [29], but by using a search strategy based on multiple pheromone information and selection by dominance.

As well, we examine configurations where the candidate set is the set of all the solutions obtained by all colonies (or the single colony) in the current iteration $\mathcal{C}^{\text{ib}}$ (*iteration-best* strategy) or since the start of the algorithm $\mathcal{C}^{\text{bf}}$ (*best-so-far* strategy).

Additionally, the concepts of $\mathcal{MM}$AS are applied to the MO-ACO algorithm for the bQAP with minor differences. Firstly, each ant deposits a constant amount of pheromone when updating the pheromone information. Hence, the pheromone limits are calculated by means of this constant value. Secondly, some configurations allow more than one ant to update the pheromone information at a time. For this reason, the upper pheromone limit (τ_{max}) is only used to initialize the pheromone information but does not restrict the maximum value that this pheromone information can take on. Finally, evaporation is done as in Eq. (3.2) on page 16.

The configurations considered in the design of a MO-ACO algorithm for the bQAP are summarized in Table 4.2. The following chapter explains the experimental setup and the performance assessment methodology used in the experiments performed with these configurations.

Class	Selection	Ph. Inform.	Weights
D	by Dominance	Single (Eq. 3.1)	—
$\mathrm{S}_{\mathrm{all}}$	by Objective	Multiple (Eq. 4.2)	In all directions
$\mathrm{S}_{\mathrm{one}}$	by Objective	Multiple (Eq. 4.2)	In one direction

Table 4.1: Classes of search strategy used by MO-ACO when applied to the bQAP. *Selection methods* are explained in Sec. 4.3; the possible definitions of multi-objective *pheromone information* are discussed in Sec. 4.2; the different alternatives in order to define the set of *weight vectors* are examined at the end of Sec. 4.2.1.

Colonies	Candidate Set	Class	LS method
Single	$\mathcal{C}^{\mathrm{ib}}$	D	PLS, BPLS_A, w/o LS
		$\mathrm{S}_{\mathrm{all}}$	W-LS, w/o LS
		$\mathrm{S}_{\mathrm{one}}$	W-LS, w/o LS
	$\mathcal{C}^{\mathrm{bf}}$	D	PLS, BPLS_A, w/o LS
		$\mathrm{S}_{\mathrm{all}}$	W-LS, w/o LS
		$\mathrm{S}_{\mathrm{one}}$	W-LS, w/o LS
Multiple	$\mathcal{C}^{\mathrm{ib}}$	D	PLS, BPLS_A, w/o LS
		$\mathrm{S}_{\mathrm{all}}$	W-LS, w/o LS
		$\mathrm{S}_{\mathrm{one}}$	W-LS, w/o LS
	$\mathcal{C}^{\mathrm{bf}}$	D	PLS, BPLS_A, w/o LS
		$\mathrm{S}_{\mathrm{all}}$	W-LS, w/o LS
		$\mathrm{S}_{\mathrm{one}}$	W-LS, w/o LS

Table 4.2: Configurations of the MO-ACO algorithm for the bQAP. The column *Colonies* refers to the *single* colony approach compared to use of *multiple* colonies. For both approaches, an iteration-best strategy ($\mathcal{C}^{\mathrm{ib}}$) or a best-so-far strategy ($\mathcal{C}^{\mathrm{bf}}$) may define the *candidate set*. For each of these possibilities, various *classes* of search strategies (either D, $\mathrm{S}_{\mathrm{all}}$ or $\mathrm{S}_{\mathrm{one}}$) will be tested. Finally, the appropriate *local search method* for each class will be applied (*PLS* or *BPLS*$_A$ for class D and *W-LS* for class S) versus the same configurations without local search (*w/o LS*).

Chapter 5

Performance Assessment

The performance assessment of a multi-objective algorithm is not an easy task. In the single objective case the quality of the outcome of an algorithm is defined with respect to the objective value. However, it is not clear how to define the quality of a Pareto set of objective vectors. Consequently, many different quality measures have been proposed in the literature. These quality measures can be classified into unary measures, which assign a quality value to a Pareto set independently from other Pareto sets, and binary measures, which compare two Pareto sets. The binary ϵ-measure indicates whether a Pareto set is better than another in terms of Pareto optimality and the number of times that an outcome obtained by one algorithm is better than an outcome obtained by the other can give an idea of their performance. Unfortunately, in many cases the outcomes may be incomparable in terms of Pareto optimality. An unary ϵ-measure can be defined in the case that the optimal Pareto set is known. This unary ϵ-measure is used to discriminate among incomparable Pareto sets. Because the Pareto optimal set is not known for the *b*QAP instances considered, we calculate a lower bound of the Pareto optimal set and use this lower bound in order to define the unary ϵ-indicator. Additionally, an Analysis of Variance (ANOVA) of the values of this unary ϵ-measure may indicate which parameters of an algorithm contribute to its overall performance and which interactions occur between these parameters.

5.1 Binary ϵ-measure

Nowadays, an important research effort in multi-objective optimization aims to clarify the performance assessment of multi-objective algorithms. In single objective problems, the quality of the outcome of an optimization algorithm is defined in terms of the scalar value of the single objective function. In contrast, in multi-objective optimization, the concept of the quality of a Pareto set is rather vague.

The relations between Pareto sets provide a solid quality classification in order to compare the outcomes of multi-objective optimization algorithms. Particularly, the relation "better" is the most general form of superior quality between two Pareto sets. Then, given that the outcome of the algorithm $\mathcal{X}$ is the Pareto set A and the outcome of algorithm $\mathcal{Y}$ is the Pareto set B, when $A \lhd B$ we can say that algorithm $\mathcal{X}$ performs better than algorithm $\mathcal{Y}$ for these two outcomes. However, the outcomes of two algorithms may be incomparable ($A \parallel B$) in terms of Pareto optimality.

Which other quality criteria can be define over Pareto sets is still an open question. Despite this fact, many different quality measures have been defined in the literature. These quality measures can be classified in *unary* measures, which assign a quality value to a Pareto set independent of other Pareto sets under consideration, and *binary* measures, which evaluate the comparison of two Pareto sets. Binary measures require to calculate $N \cdot (N - 1)$ values in order to compare N Pareto sets because for each pair of Pareto sets a binary measure returns two values.[1] On the other hand, unary measures are most commonly used in the literature because they are more efficient to calculate, i.e., only N values must be calculated to compare N Pareto sets.

Unfortunately, there is no unary measure which can indicate whether a Pareto set is better than another [58]. On the other hand, some binary measures are able to indicate whether a Pareto set is better than another. Thus, the inference power of unary measures is insufficient to achieve conclusions as solid as those that can be obtained using binary measures.

The binary ϵ-indicator gives the factor by which an approximation set is worse than another with respect to all objectives [58]. Formally, given two Pareto sets, A and B, of an arbitrary multi-objective problem with Q objectives, the binary ϵ-indicator can be calculated as

$$I_\epsilon(A, B) = \max_{b \in B} \min_{a \in A} \max_{q \in Q} \left(\frac{a_q}{b_q} \right) \tag{5.1}$$

Using this indicator we can define a binary ϵ-measure which detects whether a Pareto set is better than another as follows

$$I_\epsilon(A, B) \leq 1 \ \wedge \ I_\epsilon(B, A) > 1 \quad \Longleftrightarrow \quad A \lhd B$$

In order to compare two groups of outcomes, obtained for example by two different algorithms, we calculate the percentage of outcomes from one group that are better than an outcome from the second group using this binary ϵ-measure. We must also calculate the percentage of outcomes from the second group that are better than an outcome from the first group. The sum of these percentages is never greater than 100%. Moreover, the remainder percentage not included in the sum of those percentages corresponds to the outcomes that are incomparable or equal.

[1] For simplification, we do not take into account symmetric binary measures or combinations of unary measures because their properties are similar to the properties of unary measures [58].

5.2 Unary ϵ-measure

We would also like to distinguish between Pareto sets which are incomparable in terms of Pareto optimality. To address this issue, we define an unary ϵ-measure using a lower bound of the optimal Pareto set that gives the factor by which a Pareto set is worse than the lower bound with respect to all objectives.

When the optimal Pareto set $\mathcal{P}$ is known, an unary ϵ-indicator can be defined as

$$I_\epsilon(A) = I_\epsilon(A, \mathcal{P})$$

This indicator gives the factor by which a Pareto set A is worse that the optimal Pareto set $\mathcal{P}$ with respect to all objectives [58]. Thus, $I_\epsilon(A) \not< 1$ and $I_\epsilon(A) = 1$ implies $A = \mathcal{P}$. Although there is no unary measure which can detect whether a Pareto set is better than other, the above unary ϵ-indicator is able to detect whether a Pareto set is *not worse* than other [58]. Thus, an unary ϵ-measure can be defined as

$$I_\epsilon(A) < I_\epsilon(B) \Rightarrow B \ntrianglelefteq A$$

As well, $B \ntrianglelefteq A \iff (A \lhd B) \vee (A \parallel B) \vee (A = B)$, thus this unary ϵ-measure is consistent with the Pareto optimality criterion: if the value of I_ϵ for one Pareto set is lower than for another, then the former is better, thus we get the right answer, or both are incomparable, thus we are choosing among two incomparable sets.

Additionally, this unary ϵ-measure can be related to the worst case approximation ratio used in approximation algorithms and specially to the notion of *ϵ-approximate Pareto curve* [41]. Given a Pareto set A with $I_\epsilon(A) = 1 + \epsilon$, then there is not other Pareto set B such that for any $b \in B$ then for all $a \in A$, $b_q \leqslant (1 + \epsilon)a_q$ for some $q = 1, \ldots, Q$. Hence, this is a worst case measure and a reasonable answer to present to the decision-maker.

Nevertheless, the optimal Pareto set for the bQAP is usually not known. In this case, a lower bound $\mathcal{B}$, such that $\mathcal{B} \lhd \mathcal{P}$, can be used instead. Hence, $I_\epsilon(A) = I_\epsilon(A, \mathcal{B})$. It turns out that $I_\epsilon(A) \geqslant I_\epsilon(\mathcal{P}, \mathcal{B})$, i.e., the unary ϵ-measure is always greater or equal than the factor by which the optimal Pareto set is worse than the lower bound.

5.2.1 Lower Bound

In order to compute a lower bound to the optimal Pareto set for the bQAP, we extend the Gilmore-Lawler lower bound [25]. For the single objective QAP, the Gilmore-Lawler bound is given by the optimal objective value of an associated Linear Assignment Problem (LAP) which can be solved in cubic time (see [3] for more details).[2]

[2] Code in FORTRAN is available at QAPLIB.

In this case, we define a bi-objective LAP (bLAP) where each objective is the LAP associated to each objective of the bQAP. Then, we solve several scalarizations of the bLAP, obtaining a number of solutions which are not dominated by any Pareto optimal solution of the associated bQAP. However, these solutions are not guaranteed to dominate all the Pareto optimal solutions of the bQAP. Therefore, we added points as follows. First, we sort the objective vectors lexicographically and for each successive pair of objective vectors u, v we added another point w with coordinates $w_1 = \min\{u_1, v_1\}$ and $w_2 = \min\{u_2, v_2\}$. The resulting set is guaranteed to dominate all Pareto optimal solutions of the associated bQAP. Figure 5.1 gives a pictorial perspective of this worst case lower bound: the white points corresponds to the optimal points for several scalarizations of the bLAP and the dark points refer to the additional points. For the instances considered, we used 5000 weight vectors maximally dispersed in $[0, 1]$.

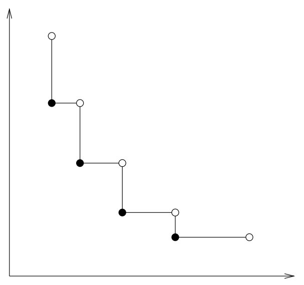
Figure 5.1: Pictorial view of the lower bound for the bQAP.

5.2.2 Analysis of Variance (ANOVA)

The performance assessment of an algorithm or a set of algorithms usually involves setting the values of certain parameters, e.g., the evaporation factor in ACO algorithms. Each experiment characterizes a certain configuration of the values of these parameters. In order to decide which algorithm or which of its possible configurations performs better than the others, it normally suffices to repeat several times each experiment, calculate the quality measure for each repetition and compare the mean values of the measure for each experiment. However, this sort of analysis does not generally explain why a certain configuration is better or worse than another. Analysis of Variance (ANOVA) is a statistical technique that may be used to find out which parameters of an algorithm are relevant to explain its performance. Furthermore, ANOVA may also indicate the relations between these parameters.

The parameters which vary depending on the experiment and may affect the performance are called *factors*. The *levels* of the factor are the different values that each factor takes on for every experiment. Finally, the quality measure used to assess performance is the *response* variable. In our analysis, the response variable is the value of the unary ϵ-measure using the lower bound and the factors are the different components of a certain configuration of MO-ACO when applied to the bQAP.

Essentially, two key question can be asked: does a factor influence the response variable? is the effect of a factor in the response variable modified by the influence of another factor? The first question is whether there is a *main effect*, whereas the second one asks for an *interaction effect*. The initial hypothesis is there is not effect, i.e., for each factor or interaction the *null hypothesis* is that it does not influence on the response variable. ANOVA tests this null hypothesis and gives the probability that it is in fact true. This probability is called a p-value. When the p-value is lower than a certain value, by convention 0.05, it is said to be statistically significant and then the null hypothesis is rejected, i.e., there is a significant effect of the factor or interaction. On the other hand, when the p-value is not statistically significant the null hypothesis cannot be rejected, thus the experimental results do not evidence any significant difference on the response variable for any level of the factor.

The results of ANOVA are expressed as a characteristic table. Each row shows the results for a certain factor or for a certain interaction. Each column shows the results of calculations involved in the test of the hypothesis. The most important element of this table is the column of the p-value.

In the case of main effects, if the null hypothesis is rejected then some levels of a factor have a different effect on the response variable than the other levels of the factor. However, ANOVA does not tell which pairs of levels are significantly different. For this purpose, all possible pairs of the means for each level can be compared using confidence intervals. The Tukey method obtains confidence intervals at the 95% for all the pairwise comparisons. Each interval indicates that there is a 95 percent of probability that the interval contains the real mean of the response variable for a certain level. Therefore, if the intervals do not themselves overlap, the means are significantly different.

For interaction effects, the means of the levels of one factor can be plotted for each level of other factor. The interaction effect is indicated by non parallel lines. However, this interaction plot neither proves whether there is a significant interaction effect nor it shows the real difference between the combinations of levels. The first issue is achieved by ANOVA. The second one is attained by plotting error bars that represents the upper and lower bounds of a 95 percent confidence interval around each mean. Thus, the Tukey method is used to calculate these intervals for each combination of levels.

The ANOVA is based on some assumptions that must be checked before performing the actual statistical analysis. Appendix A explains briefly how these assumptions are checked.

5.3 Median Attainment Surface

The probability of obtaining an arbitrary goal in the objective space during a single run of an arbitrary algorithm can be represented by an attainment function [27]. This attainment function can be estimated using data collected from several runs of the particular algorithm. The *median attainment surface* is the line that connects objective vectors assigned to an empirical frequency of 50% of being attained [27]. The median attainment surface gives a clear visual information of how the objective space is covered by the outcomes of an algorithm. Particularly, we will use median attainment surfaces to assess the similarities and differences between those configurations that are incomparable with respect to the binary ϵ-measure.

5.4 Reference Solutions

We would like to obtain reference solutions in order to compare with the outcomes of the configurations of MO-ACO studied here. However, there is no exact algorithm for the bQAP. Robust Taboo search (RoTS) [52] is among the best performing algorithms for unstructured instances of the single objective QAP and it obtains good solutions for structured instances [19, 51]. Paquete & Stützle [43] used Robust Taboo Search (RoTS) to solve several scalarizations of the bQAP using maximally dispersed weight vectors. The solutions generated by the several runs are filtered by removing dominated solutions in order to obtain a Pareto set. This algorithm is called W-RoTS.

Chapter 6

Experiments

6.1 Experimental Setup

The experimental setup involved three factors: *(i)* the search configuration, which can be of class D, $\mathrm{S}_{\mathrm{all}}$ or $\mathrm{S}_{\mathrm{one}}$; *(ii)* iteration-best versus best-so-far strategies for defining the candidate set; and *(iii)* one or multiple colonies. In addition, each combination of these factors was run with and without local search. The local search methods tested were W-LS for classes $\mathrm{S}_{\mathrm{all}}$ and $\mathrm{S}_{\mathrm{one}}$; and PLS and BPLS_A for class D. For BPLS_A we tested three different values of the upper bound limit of the archive size, $A \in \{100, 500, 1000\}$. Therefore, when local search was used, the search configurations were: class D with PLS, BPLS_{100}, BPLS_{500}, and BPLS_{1000}; class $\mathrm{S}_{\mathrm{all}}$ with W-LS and class $\mathrm{S}_{\mathrm{one}}$ with W-LS.

The total number of ants (m) was always equal to the size of the instance. Five colonies ($c = 5$) were used in the multi-colony approach, thus there were $m/5$ ants per colony. We followed the rules of $\mathcal{MM}$AS for the management of pheromones with $\alpha = 1$ and $\beta = 0$ (heuristic information is not used); $\rho = 0.9$ for the pheromone evaporation; $p_{best} = 0.05$ to derive the factor between the lower and the upper pheromone trail limits; and τ_{max} was set to the theoretically largest value [51].

The algorithms were tested on the six instances of size $n = 50$ presented in Section 2.1.1 on page 8. Three of these instances are *unstructured* and the other three are *structured* instances. These three instances were generated with $\xi \in \{0.75, 0.00, -0.75\}$, where ξ is a parameter that influences the correlation between the flow matrices.

Every experiment was run for a maximum time of 300 CPU seconds and was repeated 20 times. The algorithms were coded in `C` and the experiments were done on a computer with a Pentium III 933 MHz CPU and 512 MB RAM running under Debian Linux.

6.2 Analysis of Experimental Results

The analysis of the results is done in three phases. The first phase uses the binary ϵ-measure to detect which configurations perform clearly better. In the second phase, a lower bound of the optimal Pareto set is calculated for each instance and we perform an Analysis of Variance (ANOVA) on the values of the unary ϵ-measure with respect to the lower bound. Finally, the results are visualized using median attainment surfaces.

Additionally, we obtained reference solutions using W-RoTS. We ran W-RoTS for a total time of 300 CPU seconds for each instance, as done for each experiment of MO-ACO. Each run of the underlying taboo search algorithm was stopped after $100 \cdot n$ iterations. With this time limit, approximately 136 scalarizations could be run. These scalarizations used a set of maximally dispersed weight vectors in the interval $[0, 1]$.

6.2.1 Analysis Based on Binary ϵ-measure

The first phase of the experimental analysis allows us to obtain solid conclusions about the performance of the different configurations. The aim of this phase is to identify which configurations of MO-ACO perform clearly better than others. The binary ϵ-measure indicates whether one outcome is better than another, the latter is better than the former or they are incomparable in terms of Pareto optimality. Using this binary ϵ-measure, we compare two set of outcomes, comparing each outcome from the first set with each outcome from the second set. First, we calculate the percentage of times one outcome from the first set is better than one outcome from the second set. Next, we must also calculate the percentage of times one outcome from the second set is better than one outcome from the first set. The remainder of the sum of these two values is the percentage of times one outcome from any of the sets is incomparable or equal to one outcome from the other set.

Table 6.1 compares configurations using local search methods with configurations not using local search. Comparison are grouped according the local search methods tested: BPLS_{100}, BPLS_{500}, BPLS_{1000} and PLS for class D and W-LS for class S. Within each group, separated by a vertical line, configurations using the respective local search are compared with the same configurations not using local search (*w/o*). Each value of the two columns within each group, that is, each of the pair of values between vertical lines, is the percentage of times an outcome corresponding to one column is better than an outcome corresponding to the other column. For each pair of percentages, the first one, corresponding to configurations using a particular local search method, is always greater than 25%, while the second value, corresponding to the same configurations without local search, is always less than 0.8% with only two exceptions. Moreover, the difference between the first value minus the second one is always greater than 15% and for any instance

Type	ξ	BPLS_{100}	w/o	BPLS_{500}	w/o	BPLS_{1000}	w/o	PLS	w/o	W-LS	w/o
Un.	0.75	35.4	15.0	82.7	0.0	97.4	0.0	99.9	0.0	100.0	0.0
	0.00	99.1	0.0	99.1	0.0	99.1	0.0	98.8	0.0	99.4	0.0
	−0.75	69.9	0.0	90.5	0.0	93.0	0.0	86.1	0.0	100.0	0.0
Str.	0.75	88.4	0.0	86.8	0.0	79.3	0.0	34.3	5.7	99.5	0.0
	0.00	97.7	0.0	95.4	0.0	93.5	0.0	44.3	0.8	99.9	0.0
	−0.75	82.4	0.0	72.7	0.0	61.7	0.0	26.3	0.4	100.0	0.0

Table 6.1: Comparison of configurations using local search methods (either BPLS_{100}, BPLS_{500}, BPLS_{1000} and PLS for class D and W-LS for class S) with configurations not using local search (*w/o*). Each group of comparisons is separated by a vertical line. Each entry gives the percentage of times an outcome corresponding to one of the two columns of the group was better than an outcome corresponding to the other column.

there is always at least two local search methods for which this difference is greater than 80%. Therefore, the results show that configurations using local search methods obtain better results that configurations not using local search. Furthermore, for the structured instances (*Str.*), the three last rows of Table 6.1 show that the highest differences of percentages are obtained when using W-LS, always greater than 99%, and the bounded variant BPLS_A, always greater than 80% for at least one value of $A = \{100, 500, 1000\}$. In contrast, for PLS this difference is less than 45% for any structured instance. Thus, BPLS_A is a better local search method than PLS for configurations of class D when tackling structured instances.

Additionally, we have the intuition that the results obtained with certain configurations can be very different depending on whether local search is used. In the following, in order to show this difference, we study separately those configurations which use local search of the ones that do not use it.

Next, we compare those configurations based on scalarizations in all directions at each iteration ($\mathrm{S_{all}}$) with configurations based on scalarizations in one direction at one iteration and a different direction in the next iteration ($\mathrm{S_{one}}$). Results are summarized in Table 6.2, for configurations which use W-LS, which are of class S, and for configurations of class S which do not use any local search. Since the differences between the pairs of percentages are almost zero independently of the instance and of the use of local search, we conclude that the strategies ($\mathrm{S_{all}}$) and ($\mathrm{S_{one}}$) are mainly incomparable in terms of Pareto optimality. This incomparability, however, can be caused by two different situations: either the outcomes of configurations of classes $\mathrm{S_{all}}$ and $\mathrm{S_{one}}$ are very similar, that is, the shape of the outcomes is very similar, or the outcomes are very different but incomparable in terms of Pareto optimality, that is, the outcomes are different but for both strategies $\mathrm{S_{all}}$ and $\mathrm{S_{one}}$ the outcomes obtained using one strategy are not better than the outcomes obtained by the other. This first phase of the analysis cannot answer to this

Instance		W-LS		w/o LS	
Type	ξ	S_{all}	S_{one}	S_{all}	S_{one}
Un.	0.75	7.1	10.1	11.9	16.9
	0.00	0.7	0.3	2.9	6.0
	−0.75	0.0	0.0	2.1	2.8
Str.	0.75	0.0	0.0	4.1	4.0
	0.00	0.0	0.0	4.4	1.8
	−0.75	0.0	0.0	2.8	3.3

Table 6.2: Comparison of configurations using S_{all} with configurations using S_{one}. Each entry gives the percentage of times an outcome obtained by a configuration of that column was better than an outcome obtained by a configuration of the other column. The same comparison is made separately for configurations using W-LS and for configurations not using local search (*w/o LS*).

question. Nonetheless, we visualized these results by means of the median attainment surfaces (as explained in Section 5.3) to obtain the answer. We find out that for configurations where the only difference was the use of S_{all} or S_{one}, the outcomes were very similar. Therefore, we conclude that for the bQAP there is no difference between the approaches used by S_{all} and S_{one}.

The next question is which approach among class D and class S obtains the best outcomes. In order to have the same number of outcomes in each group, we consider only one of the search configurations of class S, particularly we consider only S_{all}.[1]

Table 6.3(a) compares configurations of class S using W-LS with configurations of class D using either BPLS_{100}, BPLS_{500}, BPLS_{1000} or PLS. In the unstructured instance with $\xi = 0$, the percentage of times an outcome obtained by class D using any local search was better that one obtained by class S using W-LS is at least 30%, while none of the outcomes obtained by class S using W-LS was better than one obtained by class D using any local search. In the case of the unstructured instance with $\xi = -0.75$, the highest difference (12%) is obtained when class D uses BPLS_{1000}. For the unstructured instance with $\xi = 0.75$, the highest difference is obtained when class D uses PLS ($32.2 - 21.6 = 10\%$), while this difference is $0.0 - 97.7 = -97.7\%$ for BPLS_{100}, $1.3 - 63.3 = -62\%$ for BPLS_{500}, and $11.6 - 30.6 = -19\%$ for BPLS_{1000}, where the negative percentage means that class S using W-LS was better than class D using that particular local search. In summary, for the unstructured instances the results obtained by class D improve as the upper bound A on the archive size of BPLS_A increases, and class D using PLS or BPLS_{1000} is slightly better than class S using W-LS.[2]

[1] As explained in the previous paragraph, the results are very similar if S_{one} is considered instead of S_{all}.

[2] Actually, PLS is equivalent to BPLS_A when $A = \infty$.

Type	ξ	D BPLS$_{100}$	S W-LS	D BPLS$_{500}$	S W-LS	D BPLS$_{1000}$	S W-LS	D PLS	S W-LS
Un.	0.75	0.0	97.7	1.3	63.3	11.6	30.6	32.2	21.6
	0.00	40.7	0.0	46.1	0.0	45.7	0.0	30.2	0.0
	−0.75	0.1	0.0	8.9	0.0	12.0	0.0	1.3	0.0
Str.	0.75	0.0	2.4	0.0	3.7	0.0	16.6	0.0	83.6
	0.00	0.0	0.0	0.0	1.0	0.0	2.6	0.0	63.7
	−0.75	0.0	1.1	0.0	3.0	0.0	8.9	0.0	71.2

(a) D *vs.* S with LS

Type	ξ	D w/o LS	S w/o LS
Un.	0.75	59.5	0.1
	0.00	52.6	0.0
	−0.75	27.7	1.0
Str.	0.75	29.2	0.5
	0.00	25.9	0.1
	−0.75	28.8	0.1

(b) D *vs.* S without LS

Table 6.3: Comparison of configurations of class D with configurations of class S. Within each group delimited by vertical lines, each entry gives the percentage of times an outcome obtained by a configuration of that column was better than an outcome obtained by a configuration of the other column. The same comparison is made separately for *(a)* configurations using local search, W-LS for class S and either PLS or BPLS$_A$ with $A = \{100, 500, 1000\}$ for class D, and *(b)* configurations not using local search (w/o LS).

With regard to the structured instances, the results are completely different: the results obtained by class D improve as the size A of the archive in BPLS$_A$ decreases. For all the values of ξ, at least 63.7% of times class S using W-LS was better than class D using PLS, while if class D uses BPLS$_{100}$ that value is never greater than 2.4%. Thus, in the case of structured instances, class D with PLS is outperformed by class S with W-LS, but if BPLS$_{100}$ is used instead of PLS, then the outcomes obtained by class D and class S are mainly incomparable.

When local search is not used, Table 6.3(b) shows that in all the six instances the percentage of times class D was better than class S is always greater than 25%, while the percentage of times class S was better is almost zero. Therefore, configurations of class D are slightly better than configurations of class S when they do not use any local search method.

Since results vary remarkably depending on which local search method is

Instance		with LS		w/o LS	
Type	ξ	*ib*	*bf*	*ib*	*bf*
Un.	0.75	8.5	17.9	3.5	51.1
	0.00	0.0	25.2	0.1	54.9
	−0.75	0.0	0.1	0.0	34.2
Str.	0.75	5.6	2.2	3.1	31.6
	0.00	1.1	0.6	0.7	24.4
	−0.75	2.1	3.6	0.1	26.2

Table 6.4: Comparison of configurations using iteration-best (*ib*) with configurations using best-so-far (*bf*) strategies. Each entry gives the percentage of times an outcome obtained by a configuration of that column was better than an outcome obtained by a configuration of the other column. The same comparison is made separately for configurations not using local search (w/o LS) and for configurations using local search (with LS). The local search methods considered in this comparison are W-LS for class S_{all}, $BPLS_{1000}$ for class D on unstructured (*Un.*) instances, and $BPLS_{100}$ for class D on structured (*Str.*) instances.

used with class D, we will consider only one local search method for configurations of class D in the analysis of the remaining parameters: the strategy used to define the candidate set (either iteration-best or best-so-far strategies) and the number of colonies $c = \{1, 5\}$. In particular, we will concentrate on $BPLS_{1000}$ in the case of the unstructured instances and on $BPLS_{100}$ for the structured ones, because they are the best overall local search methods to use in combination with class D according to Table 6.3(a).

Turning now to the strategy used to define the candidate set from which solutions used to update the pheromone information are taken, we studied two strategies: iteration-best (*ib*) and best-so-far (*bf*). Table 6.4 shows that for the unstructured instances the maximum differences of percentages are obtained by best-so-far for $\xi = 0.75$, where the difference is $8.5 - 17.9 = -9.4\%$, and for $\xi = 0$, where the difference is $0.0 - 25.2 = -25.2\%$; while for $\xi = -0.75$ and for the structured instances the difference is almost zero. Therefore, when local search is used, best-so-far is slightly better than iteration-best for unstructured instances with positive and zero correlation, while there is no clear difference between these two strategies in the case of the unstructured instance with negative correlation and the three structured instances. In contrast, when local search is not used, the best results are obtained by best-so-far with a difference of at least 24% for any of the instances.

With regard to the number of colonies, Table 6.5 compares using a single colony approach ($c = 1$) with using a multi-colony approach ($c = 5$), for configurations using local search methods and not using local search methods. When using local search methods, the only remarkable difference is in the unstructured instance with high positive correlation where using a single colony is better than using five colonies with a difference of percentages of

Instance		with LS		w/o LS	
Type	ξ	$c = 1$	$c = 5$	$c = 1$	$c = 5$
Un.	0.75	42.2	16.4	55.0	2.6
	0.00	2.6	0.9	38.7	4.9
	−0.75	3.4	1.6	12.4	12.1
Str.	0.75	0.0	4.5	37.4	4.9
	0.00	0.0	0.0	26.2	3.8
	−0.75	0.0	1.6	17.2	1.6

Table 6.5: Comparison of configurations using one colony ($c = 1$) with configurations using five colonies ($c = 5$). Each entry gives the percentage of times an outcome obtained by a configuration of that column was better than an outcome obtained by a configuration of the other column. The same comparison is made separately for configurations not using local search (w/o LS) and for configurations using local search (with LS). The local search methods considered in this comparison are W-LS for class S_{all}, BPLS_{1000} for class D on unstructured (*Un.*) instances, and BPLS_{100} for class D on structured (*Str.*) instances.

$42.2 - 16.4 = 25.8\%$. In contrast, when local search methods are not used, there are differences of percentages which are greater than 30%, in favor of the single colony approach, for unstructured instances with $\xi = 0.75$ and $\xi = 0$, and for the structured instance with $\xi = 0.75$.

In summary, this phase of the analysis strongly indicates that any of the studied configurations of MO-ACO that uses local search methods outperforms any configurations not using local search methods (Table 6.1). Additionally, the results obtained by configurations of class D vary depending on whether PLS or BPLS_A is used, and on the particular value of the upper bound A of the size of the archive (Table 6.3). Particularly, for unstructured instances the outcomes obtained by configurations of class D are as good as those obtained by configurations of class S with W-LS only when class D is combined with PLS for the instance with $\xi = 0.75$, and either with BPLS_{1000} or with BPLS_{500} in the case of $\xi = 0.00$ and $\xi = -0.75$. As well, for $\xi = 0.00$, class D combined with any of the bounded variants is slightly better than class S. In the case of structured instances, the best results are obtained when class D is combined with BPLS_{100}. This analysis showed also that there is no clear difference between using configurations of class S_{all} and using S_{one} (Table 6.2). However, this analysis was not able to tell whether the two approaches are equivalent or their outcomes are incomparable in terms of Pareto optimality.

In addition, we analyzed the results of the same configurations when they do not use any local search method, in order to show that the conclusions would be different if the use of local search is not taken into account from the beginning of the analysis. When local search is not considered, the configurations of class D are better than configurations of class S (Table 6.3(b)), and using a best-so-far strategy (Table 6.4) and one colony (Table 6.5) are also advantageous. By comparison with the previous analysis, it is obvious that these conclusions are no longer valid when local search is applied.

The analysis based on the binary ϵ-measure shows that many outcomes are incomparable in terms of Pareto optimality. In the next phase of our analysis we use the unary ϵ-measure and the Analysis of Variance (ANOVA) in order to choose between configurations that produce outcomes that are mostly incomparable in terms of Pareto optimality.

6.2.2 Analysis Using Unary ϵ-measure and ANOVA

The second phase of our analysis aims to find differences between the configurations established as incomparable in the previous phase. Moreover, the usage of an statistical analysis tool like Analysis of Variance (ANOVA), allows us to find interactions between the different parameters of the MO-ACO algorithm. However, the conclusions obtained by this analysis are neither as general nor as solid as those obtained when using the binary ϵ-measure, and the results of the analysis depend in a high degree on the quality of the lower bound, i.e., on how well the lower bound represents the optimal Pareto set. A good example of this issue is given by the lower bound for the structured instances. Figure 6.1 shows the lower bounds calculated for each instance and the reference solutions obtained using W-RoTS. In the case of the structured instances, the lower bound does not have enough quality in order to perform this analysis. Moreover, the assumptions needed to perform ANOVA are not fulfilled for the structured instances (see Fig. A.3). Therefore, this second phase of the analysis only considers the unstructured instances.

Unstructured instances are analyzed for each value of ξ. Furthermore, since we already know that any configuration that uses local search is better than any other without local search, we would only need to analyze configurations with local search. However, we will also analyze separately configurations that do not use local search in order to show that the conclusions obtained without local search are not valid when local search is used, and, hence, local search cannot be simply "added" to the algorithm *a posteriori*.

As a first step in this analysis, the value of the unary ϵ-measure is calculated for each outcome using the lower bound of the respective instance. Next, these values are analyzed using ANOVA. The requirements of ANOVA are checked by means of the appropriate plots, which are shown in Appendix A. The ANOVA tables tell us which factors are statistically significant, i.e., which factors actually affect the value of the unary ϵ-measure. The factors taken into account in the ANOVA analysis are the same as those considered previously, with the exception of the class of configuration, where each of the local search methods tested when using class D is considered as a different class. Therefore, the levels of the factor *Class* are class D with BPLS_{100} (BPLS_{100}), class D with BPLS_{500} (BPLS_{500}), class D with BPLS_{1000} (BPLS_{1000}), class D

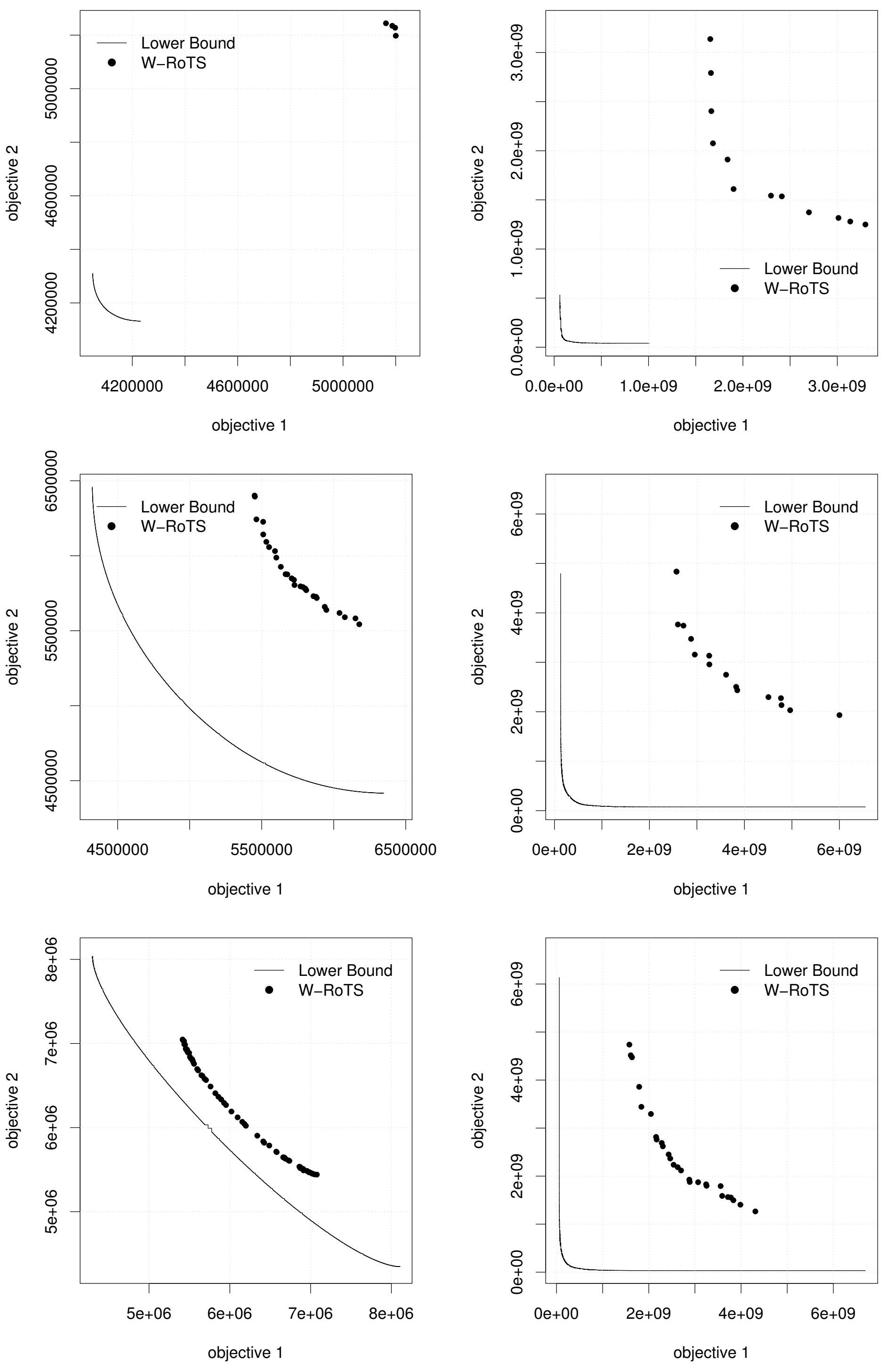

Figure 6.1: Lower bounds for unstructured (left) and structured (right) instances with $\xi = \{0.75$ (top), 0.0 (center) and -0.75 (bottom)$\}$. The objective vectors obtained by W-RoTS are also plotted (points) for comparison.

with PLS (PLS), class S_{all} with W-LS (S_{all}), and class S_{one} with W-LS (S_{one}). The other factors are the strategy followed to define the candidate set ($\mathcal{C}$ *set*) from which the solutions used to update the pheromone information are taken, which may be iteration-best (*ib*) or best-so-far (*bf*); and the number of colonies (*Colonies*), which can be 1 or 5.

An example of ANOVA table is Table 6.6, where each of the first three rows is a main factor and the following three rows are the interactions of these main factors, denoted by the symbol "$\times$". We are only interested in the column labeled p-value because when this value is lower than 0.05 then the main factor or interaction of that row shows a statistically significant effect on the value of the unary ϵ-measure. In order to easily identify the statistically significant effect, the symbol "***" denotes that the p-value is in the interval $(0, 0.001)$, "**" denotes it is in $(0.001, 0.01)$, "*" denotes it is in $(0.01, 0.05)$, "." denotes it is in $(0.05, 0.1)$, and p-values greater than 0.1 are denoted by a blank space. The other values are calculated during ANOVA and are given for informative purpose only. This output format of the Analysis of Variance is given by R [44], an environment for statistical computing.

In order to evidence which levels of a factor show a statistically significant difference and which level is the one with the lowest value of the unary ϵ-measure, we plot the mean value of each level (or combination of levels for interactions) and a confidence interval around this mean. If two intervals do not overlap, there is a 95% confidence that the two levels are actually different, whereas if the intervals overlap then there is no statistically significant reason to reject the hypothesis that the two levels have the same effect on the value of the unary ϵ-measure.

Table 6.6 shows the results of ANOVA for the unstructured instance with $\xi = 0.75$. There are two statistically significant interaction effects, *Class* $\times$ *Colonies* and $\mathcal{C}$ *set* $\times$ *Colonies*, that comprise all the main effects. These two interaction effects are plotted in Fig. 6.2. The plot on the left shows that the best value of the unary ϵ-measure is obtained by class S using 1 colony or by class D with PLS using 5 colonies. In the plot on the right we observe that the use of a candidate set based on an iteration-best strategy combined with 5 colonies gives worse results than the other alternatives.

For $\xi = 0.00$, the results of the ANOVA are quite different. In this case, as shown in Table 6.7, there is an interaction effect between the class of search strategy and the candidate set used, *Class* $\times$ $\mathcal{C}$ *set*. Additionally, there is a main effect of the number of colonies. This main effect shows that those configurations with one colony achieve better values of the unary ϵ-measure than configurations with five colonies (Fig. 6.3). The interaction effect, plotted in Fig. 6.4, shows that for configurations of class S there is a significant difference between using an iteration-best candidate set (*ib*) or a best-so-far strategy (*bf*). In contrast to the instance with $\xi = 0.75$, in this case it is better to use the best-so-far strategy.

In the negative correlated instance ($\xi = -0.75$), the interaction effect

	Df	Sum Sq	Mean Sq	F value	p-value	
Class	5	0.033008	0.006602	226.7388	< 2.2e-16	***
$\mathcal{C}$ set	1	0.000157	0.000157	5.3764	0.02085	*
Colonies	1	0.000478	0.000478	16.4332	5.915e-05	***
Class × $\mathcal{C}$ set	5	0.000279	0.000056	1.9161	0.09027	.
Class × Colonies	5	0.002618	0.000524	17.9807	2.645e-16	***
$\mathcal{C}$ set × Colonies	1	0.001007	0.001007	34.5702	7.900e-09	***
Residuals	461	0.013422	0.000029			

Table 6.6: ANOVA table for configurations using local search applied to an unstructured instance of size 50 with $\xi = 0.75$.

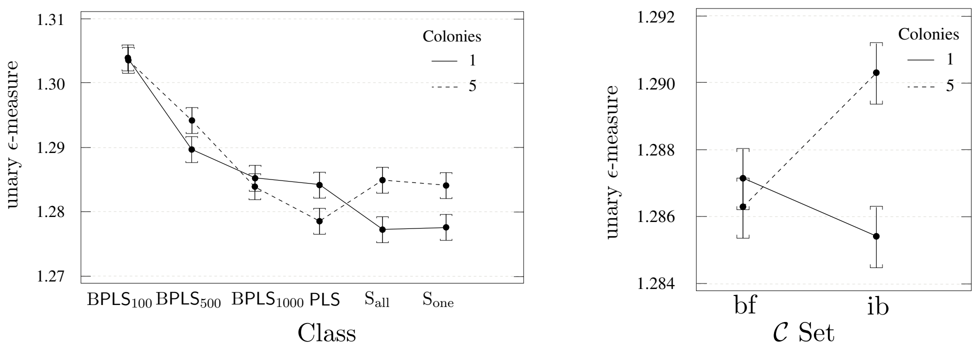

Figure 6.2: Interactions for configurations using local search applied to an unstructured instance of size 50 with $\xi = 0.75$.

	Df	Sum Sq	Mean Sq	F value	p-value	
Class	5	0.000409	0.000082	12.1967	4.069e-11	***
$\mathcal{C}$ set	1	0.001420	0.001420	211.8403	< 2.2e-16	***
Colonies	1	0.000054	0.000054	7.9874	0.004915	**
Class × $\mathcal{C}$ set	5	0.001754	0.000351	52.3159	< 2.2e-16	***
Class × Colonies	5	0.000030	0.000006	0.8992	0.481370	
$\mathcal{C}$ set × Colonies	1	0.000001	0.000001	0.1006	0.751218	
Residuals	461	0.003091	0.000007			

Table 6.7: ANOVA table for configurations using local search applied to an unstructured instance of size 50 with $\xi = 0.00$.

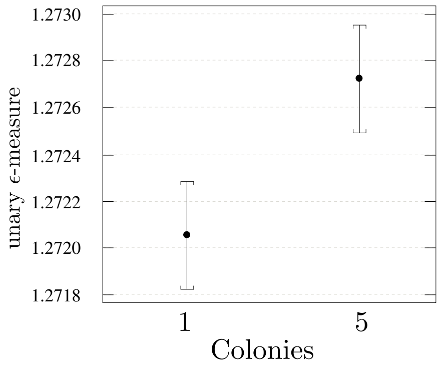

Figure 6.3: Main effect for configurations using local search applied to an unstructured instance of size 50 with $\xi = 0.00$.

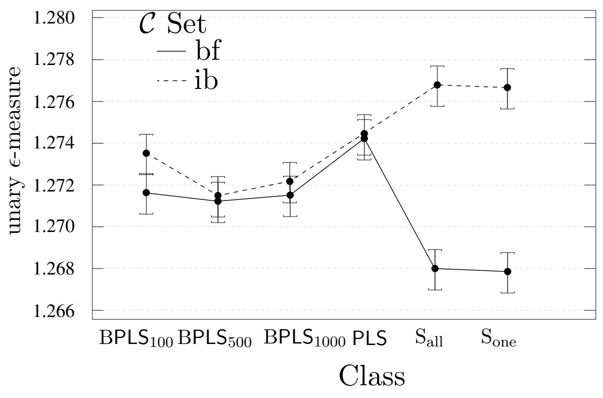

Figure 6.4: Interaction effect for configurations using local search applied to an unstructured instance of size 50 with $\xi = 0.00$.

	Df	Sum Sq	Mean Sq	F value	p-value	
Class	5	0.006822	0.001364	120.1964	<2e-16	***
$\mathcal{C}$ set	1	0.000840	0.000840	74.0430	<2e-16	***
Colonies	1	0.000013	0.000013	1.1472	0.2847	
Class × $\mathcal{C}$ set	5	0.002300	0.000460	40.5326	<2e-16	***
Class × Colonies	5	0.000051	0.000010	0.9053	0.4774	
$\mathcal{C}$ set × Colonies	1	0.000002	0.000002	0.1351	0.7134	
Residuals	461	0.005233	0.000011			

Table 6.8: ANOVA table for configurations using local search applied to an unstructured instance of size 50 with $\xi = -0.75$.

shown by ANOVA (Table 6.8), *Class* × $\mathcal{C}$ *set*, comprises the search strategy class and the type of candidate set, whereas there is no statistically significant effect that involves the number of colonies (*Colonies*). Thus, there is no significant difference between using one or five colonies. The interaction plot (Fig. 6.5) shows that configurations of class S should use the best-so-far candidate set in order to obtain the lowest values of the unary ϵ-measure.

We already know from the previous phase of our analysis that the use of local search produces better results and, hence, we do not really need to perform the ANOVA analysis for configurations not using local search. However, we briefly summarize the results of ANOVA: for each of the three unstructured instances, we report its ANOVA table and the best performing configuration. For the instance with high positive correlation ($\xi = 0.75$), as shown in Table 6.9, all interaction factors are significant. The best configuration uses a search configuration of class D with one colony and a best-so-far candidate set. For the instance with zero correlation ($\xi = 0$), there is no interaction effect between the search configuration and the number of colonies

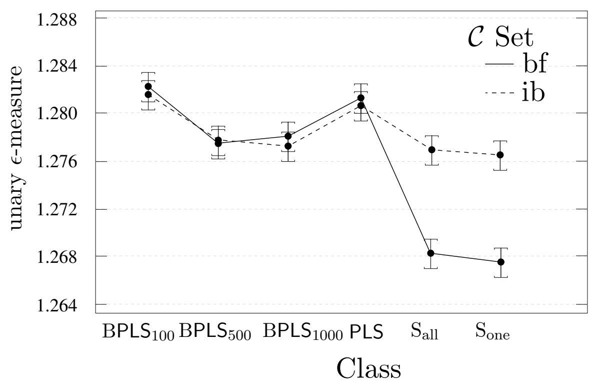

Figure 6.5: Interaction effect for configurations using local search applied to an unstructured instance of size 50 with $\xi = -0.75$.

(Table 6.10). Here, the best results are obtained also with a search configuration of class D with a best-so-far strategy but using five colonies. In the negatively correlated instance ($\xi = -0.75$), all interaction factors are significant (Table 6.11). The best configuration uses a best-so-far strategy and five colonies with a search configuration of class S (either S_{all} or S_{one}).

These results show that the conclusions obtained when local search is not considered are very different from the results obtained when using local search and, therefore, local search cannot be simply "added" after choosing the best parameters.

6.3 Visualizing the results using Median Attainment Surfaces

The last phase of the analysis gives a visual representation of the conclusions obtained in the previous phases by means of *median attainment surfaces*. The median attainment surface connects objective vectors that are attained by a certain configuration with a frequency of 50% [27]. The objective vectors obtained by W-RoTS are also plotted for reference.

For unstructured instances, Figure 6.6 compares the best configurations, according to the previous analysis, when using local search and without local search. Firstly, for all the values of ξ, the use of local search is clearly advantageous. Secondly, for $\xi = \{0.00, -0.75\}$, there is a clear difference between using class D and class S but they are still incomparable. Configurations of class D obtain objective vectors better with respect to both objectives, whereas configurations of class S obtain objective vectors better with respect to each objective. Informally, configurations of class D obtain results closer to the "center" of the optimal Pareto set, whereas configurations of class S obtain better results with respect to the "tails".

Turning to structured instances, only configurations using local search are

	Df	Sum Sq	Mean Sq	F value	p-value	
Class	2	0.016877	0.008438	237.193	< 2.2e-16	***
$\mathcal{C}$ set	1	0.157272	0.157272	4420.779	< 2.2e-16	***
Colonies	1	0.168395	0.168395	4733.436	< 2.2e-16	***
Class × $\mathcal{C}$ set	2	0.002037	0.001018	28.627	7.910e-12	***
Class × Colonies	2	0.000747	0.000373	10.493	4.354e-05	***
$\mathcal{C}$ set × Colonies	1	0.159925	0.159925	4495.356	< 2.2e-16	***
Residuals	230	0.008182	0.000036			

Table 6.9: ANOVA table for configurations not using local search applied to an unstructured instance of size 50 with $\xi = 0.75$

	Df	Sum Sq	Mean Sq	F value	p-value	
Class	2	0.013684	0.006842	191.793	< 2.2e-16	***
$\mathcal{C}$ set	1	0.120641	0.120641	3381.698	< 2.2e-16	***
Colonies	1	0.037942	0.037942	1063.569	< 2.2e-16	***
Class × $\mathcal{C}$ set	2	0.000764	0.000382	10.704	3.589e-05	***
Class × Colonies	2	0.000153	0.000076	2.142	0.1198	
$\mathcal{C}$ set × Colonies	1	0.067171	0.067171	1882.878	< 2.2e-16	***
Residuals	230	0.008205	0.000036			

Table 6.10: ANOVA table for configurations not using local search applied to an unstructured instance of size 50 with $\xi = 0.00$

	Df	Sum Sq	Mean Sq	F value	p-value	
Class	2	0.002611	0.001305	15.1781	6.432e-07	***
$\mathcal{C}$ set	1	0.188601	0.188601	2193.0117	< 2.2e-16	***
Colonies	1	0.001282	0.001282	14.9078	0.0001468	***
Class × $\mathcal{C}$ set	2	0.014371	0.007186	83.5539	< 2.2e-16	***
Class × Colonies	2	0.001243	0.000621	7.2257	0.0009048	***
$\mathcal{C}$ set × Colonies	1	0.006664	0.006664	77.4869	3.259e-16	***
Residuals	230	0.019780	0.000086			

Table 6.11: ANOVA table for configurations not using local search applied to an unstructured instance of size 50 with $\xi = -0.75$

considered and, specifically, we compare the configurations of class D using BPLS$_{100}$ and class S$_{\text{all}}$ using W-LS (Fig. 6.7).[3] One interesting result is that all these configurations are better than W-RoTS. As well, the difference between configurations of class D and S is still present, but it is not as strong as before: configurations of class S are slightly better with respect to the "tails", whereas configurations of class D obtain a few better objective vectors in the "center". The other settings do not influence clearly the quality of the outcomes.

Finally, for the unstructured instances, the shapes of median attainment surfaces vary depending on the correlation parameter ξ given to the instance generator. However, in the case of the structured instances this is no longer true. As well, from our previous analysis we concluded that for unstructured instances the results depend on the value of ξ, whereas for structured instances the results are similar for different values of ξ. In order to explain these results, we must remember that the value of ξ is a parameter given to the instance generator that induces a correlation between the flow matrices, what should also result in different correlation between the values of the objective vectors. We determined the empirical correlation between objectives through samples of 1000 random solutions generated for each instance. Table 6.12 shows that in the case of unstructured instances, the parameter ξ induces a clear correlation between the objectives, whereas there is no clear correlation in the case of structured instances. This difference is probably because of the fact that in structured instances there are many zero entries in the flow matrices. Thus, we conclude that the value of ξ does not affect clearly the correlation between objectives in the structured instances and, hence, the performance of the different configurations of MOACO on structured instances is independent of the value of ξ.

Type	ξ	*corr*
Unstructured	0.75	0.90
	0.00	−0.01
	−0.75	−0.90
Structured	0.75	0.23
	0.00	0.03
	−0.75	−0.08

Table 6.12: Correlation parameter (ξ) and empirical correlation (*corr*) for the *b*QAP instances with size 50.

[3] The results obtained with S$_{\text{one}}$ are similar to those obtained using S$_{\text{all}}$.

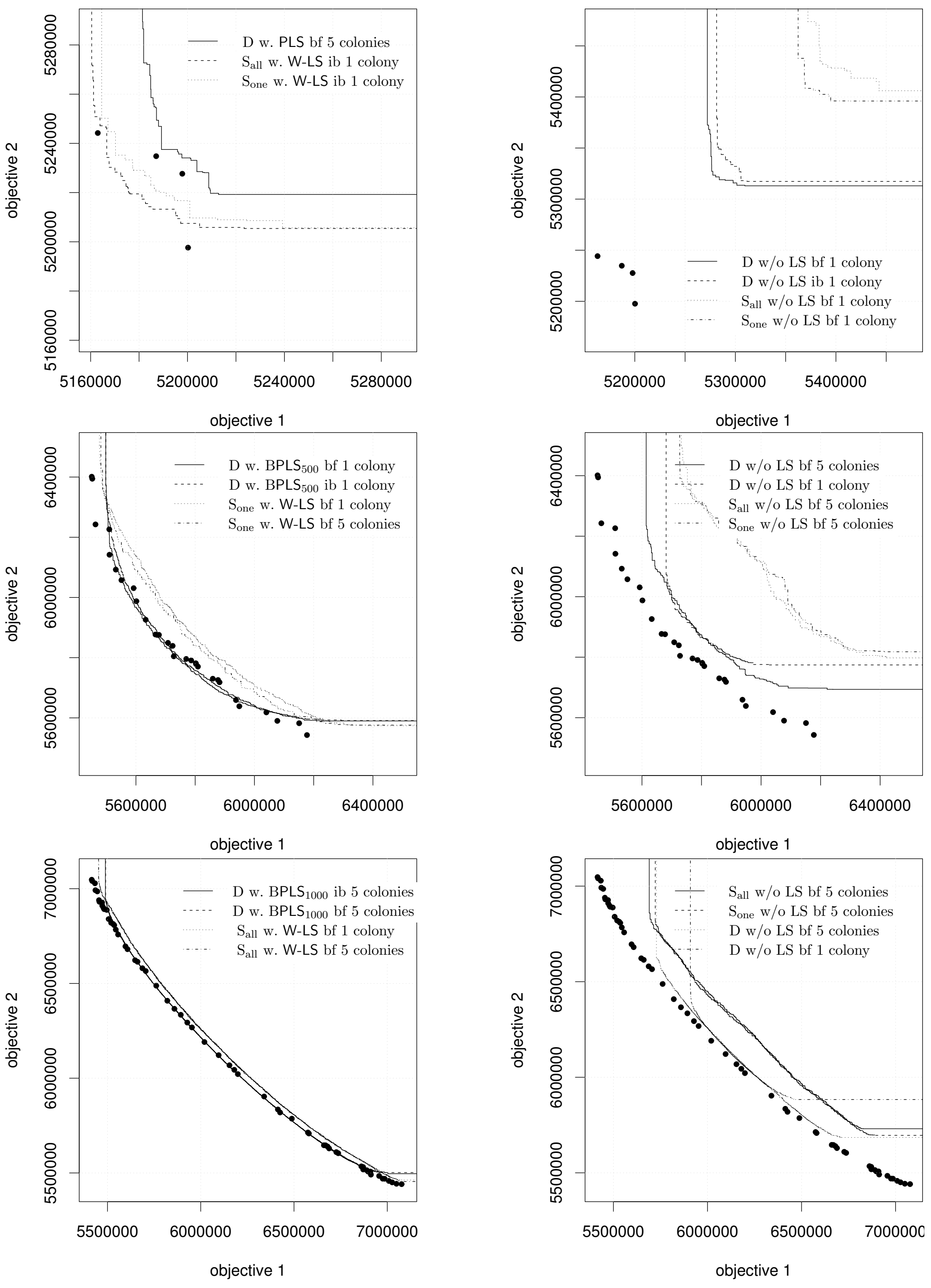

Figure 6.6: Median attainment surfaces obtained for unstructured instances of size 50 and ξ of 0.75 (top), 0.0 (center) and -0.75 (bottom) using local search methods (left column) and without local search (right column). In addition, the objective vectors obtained by W-RoTS are plotted (points). (See text for details)

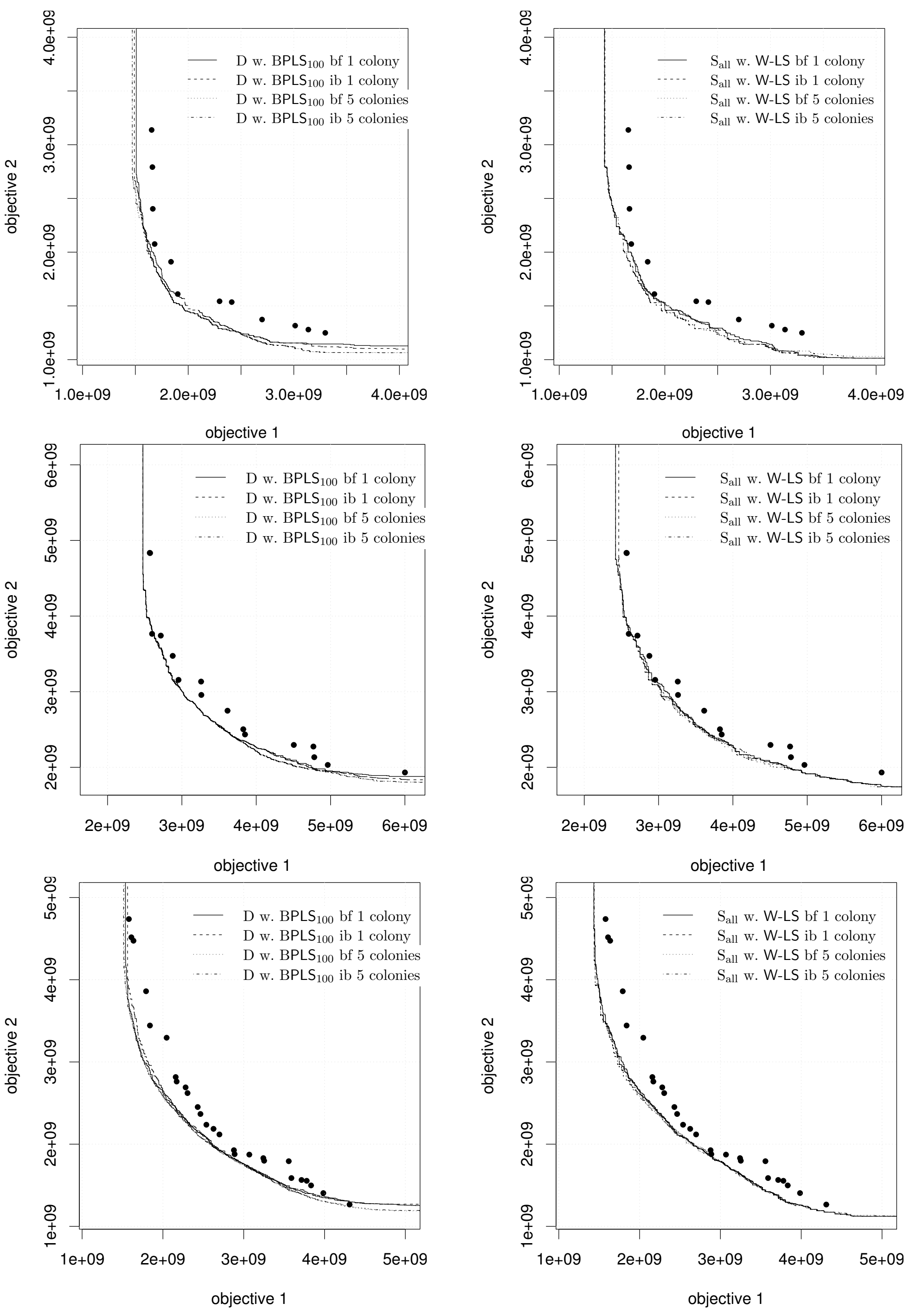

Figure 6.7: Median attainment surfaces obtained for structured instances of size 50 and ξ of 0.75 (top), 0.0 (center) and -0.75 (bottom) when using search configuration of class D with BPLS_{100} (left column) and class $\mathrm{S}_{\mathrm{all}}$ with W-LS (right column). In addition, the objective vectors obtained by W-RoTS are plotted (points). (See text for details)

Chapter 7

Conclusions

7.1 Multi-objective ACO

We have examined several alternatives in order to design an ACO algorithm to tackle MOCO problems. These alternatives have been explained as configurations of a general MO-ACO algorithm that can be adapted as needed. These configurations can be related to the two essential search strategies used in MOCO, i.e., based on dominance criterion and based on scalarizations of the objective vector. As well, we have studied the issue of defining multi-objective pheromone information. An important observation is that two different objectives may define the solution components either in a similar way or in an essentially different manner. Hence, the pheromone information for these objectives may be equivalent or essentially different. We have examined these possibilities and how to solve the problems associated with each of them. Furthermore, the computational resources required by each of these possibilities have been discussed. The result is such that these resources not only depend on the number of objectives of the particular problem being considered but also on how these objectives define the solutions components. In addition, we have considered the use of multiple colonies and how the previous alternatives can be used with a multi-colony approach.

It is also clear from our examination that the knowledge available of single objective ACO can be applied to design MO-ACO algorithms, simply by using the best performing ACO algorithms as an underlying algorithm. An example of this is the use of $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System in order to tackle the bQAP. Furthermore, the best-so-far and iteration-best strategies known in single objective ACO have been adapted to the multi-objective context.

Finally, we have examined local search methods which can be used in combination with MO-ACO algorithms. W-LS and PLS have already been proposed in the literature and both are multi-objective extensions of local search methods for single objective problems. BPLS$_A$ is a variant of PLS which uses bounded archiving techniques known from Multi-Objective Evolutionary Algorithms. All of these local search methods can be adapted to several MOCO problems.

To conclude, we have proposed an approach using ACO algorithms for MOCO problems which combines the concepts of multi-objective optimization and the knowledge of ACO for single objective optimization. As well, this approach embraces many concepts already proposed in the literature in order to deal with MOCO problems using ACO algorithms. The results of this examination are presented as a general, modular and highly configurable MO-ACO algorithm.

7.2 MO-ACO Applied to the *b*QAP

It is clear from our results that the use of local search methods is essential to obtain the best performance with MO-ACO algorithms when applied to the *b*QAP. Moreover, there are evident differences in the effect of the other parameters of MO-ACO depending on whether local search is used or not. Therefore, in order to design the best configuration of MO-ACO for the *b*QAP, the use of local search must be taken into account from the very first step of the design process. As well, when local search is not considered, the remaining parameters, i.e., the number of colonies and the type of candidate set, gain in importance. From this observation, we would say that the use of local search has an attenuation effect on the influence of the other parameters.

We also reported that for structured instances, the size of local optima Pareto sets is much larger than in the case of unstructured instances, i.e., the number of solutions that are mutually nondominated is much larger on structured instances than on unstructured ones. Thus, any method that aims to obtain the whole optimal Pareto set becomes infeasible. Furthermore, local search methods which stop when a locally optimal Pareto set is found, as PLS does, will need high computation times and huge memory requirements. We showed that the solution to this problem is the use of bounded archiving methods; e.g., BPLS$_A$ performed clearly better than PLS on structured instances. Because of its lower computation times, local search methods using bounded archiving techniques are particularly suitable for combination with MO-ACO. Therefore, for the unstructured instances with zero or negative correlation, BPLS$_A$ also obtains better Pareto sets than PLS.

Another observation is that the search configuration is the second-most important factor to influence the performance of the algorithm. First, there is no evident difference between using a search configuration based on scalarizations in all directions at each iteration (class $\mathrm{S}_{\mathrm{all}}$) and based on one scalarization in one direction at one iteration and another slightly different scalarization in the next iteration (class $\mathrm{S}_{\mathrm{one}}$). From the first phase of our analysis, we concluded that the outcomes obtained using these two search configurations were incomparable in terms of Pareto optimality, i.e., the outcomes obtained using one of these search configurations were not clearly better than the outcomes using the other. Additionally, the attainment surfaces showed that the outcomes of these approaches are very similar to each other. Hence, we conclude that for the instances of the bQAP and the parameters studied here, $\mathrm{S}_{\mathrm{all}}$ and $\mathrm{S}_{\mathrm{one}}$ are to a large extent equivalent.

In contrast, there is a clear difference between using a search configuration based on scalarizations (class S) and based on dominance criteria (class D). Both approaches with the appropriate parameters achieve high quality results, mostly incomparable in terms of Pareto optimality. However, their outcomes are quite different. The search configurations based on scalarizations obtain results closer to the "tails" of the optimal Pareto set, where solutions are good with respect to only one objective. In contrast, the outcomes of the search configurations based on dominance criteria are closer to the "center" of the optimal Pareto set, where solutions are good with respect to both objectives at the same time.

With regard to the unary ϵ-measure, the best configuration of MO-ACO depends heavily on the correlation between the objectives. In general terms, when the correlation decreases, it seems advantageous to exploit the best solutions found during the run of the algorithms using the best-so-far candidate set to update the pheromone information. Therefore, the best solutions found should be exploited in order to improve the worst-case performance.

Finally, it is clear from these initial experiments that for the bQAP, MO-ACO algorithms are a competitive approach when compared with high performing algorithms like W-RoTS, particularly on structured instances and negatively correlated unstructured instances. Nevertheless, there are many possibilities for improving MO-ACO algorithms.

7.3 Future Research

The future development of MO-ACO will depend on the ongoing research on both Ant Colony Optimization and Multi-Objective Optimization. With regard to ACO, different underlying ACO algorithms can be considered in place of $\mathcal{MM}$AS. Hence, the research on ACO for single objective problems may influence the application of MO-ACO to multi-objective variants of these problems. As well, the research on Multi-Objective Optimization will influence the future of MO-ACO. An example of this would be the recently proposed improvements on the run-time complexity of procedures used by many multi-objective meta-heuristics [30]. Consequently, the computation time of MO-ACO may be reduced using these improved procedures.

There are also particular issues solely concerning MO-ACO algorithms. For instance, in order to tackle problems with more than two objectives ($Q > 2$) several aspects must be taken into account. Firstly, the definition of the weight vectors is not as straightforward as it is with two objectives. This problem has been previously investigated in the multi-objective optimization literature [47], thus these approaches should be adapted to MO-ACO algorithms. Secondly, the "update by region" strategy for multiple colonies cannot be applied to more than two objectives. Although the "update by origin" can be used instead, this strategy does not force the colonies to focus on different regions of the objective space. Therefore, there is a need for a sound procedure which *(i)* distributes the ants allowed to update the pheromone information among the different colonies; *(ii)* forces the colonies to focus on different regions of the objective space; and *(iii)* can be used with an arbitrary number of objectives. Lastly, a further promising research direction is the use of heterogeneous configurations to combine search strategies based on scalarizations of the objective vector and on dominance relations. The goal of these heterogeneous configurations of MO-ACO is to obtain Pareto sets that are good in the "tails" and in the "center" at the same time.

Given the importance of local search methods for MO-ACO, more research is needed to develop multi-objective local search methods to combine with MO-ACO algorithms. As well, known local search methods can be adapted to be used by MO-ACO. For instance, we believe that the results obtained here for the unstructured instances of the bQAP can be improved simply by using W-RoTS instead of W-LS in the local search phase of the ACO algorithm.

In conclusion, this work could be extended by studying MO-ACO applied to other MOCO problems. Because of its modularity, the application of MO-ACO to various MOCO problems is a straightforward procedure.

Appendix A

ANOVA Assumptions

In the Analysis of Variance (ANOVA) [6], the response variable is described by the factors and interactions considered, i.e., there is a model for the factors and their interactions that approximates the response variable. The differences between the expected values of the response variable and the real values obtained in the experiments are modeled as random noise error variables and are described as *residuals*. Therefore, ANOVA assumes that the error variables *(i)* are independent, *(ii)* have constant variance for each combination of levels of the factors (homoschedasticity), and *(iii)* have a normal distribution (normality).

Before proceeding to check these assumptions, it is advisable to check for *outliers*. An outlier is an unusual result, i.e., a result much larger or much smaller than what is expected, and it is usually caused by some error or unusual situation while the experiment was performed. A few outliers are expected, whereas too many outliers will invalidate the results of ANOVA. One measure used to identify outliers is *Cook's distance*.

In the next step, the independence assumption is checked because the checks for homoschedasticity and normality assume that the error variables are independent. The independence of the error variables is checked by plotting the standardized residuals[1] for each observation. When the independence assumption is satisfied, the residuals are randomly distributed around zero with no discernible pattern.

To check the homoschedasticity of the error variables, the standardized residuals are plotted against the fitted values of the response variable. The homoschedasticity assumption is not satisfied if the residuals exhibit a clearly nonrandom pattern around zero, being too often positive for some levels of the response variable and too often negative for others. Hence, the model assumed by ANOVA does not adequately describe the observed values of the response variable. The most common pattern resembles a megaphone in shape (or its mirror image) and occurs when the error variance increases

[1] Standardized residuals are obtained by dividing the residuals by their standard deviation.

as the mean response increases. Generally, a transformation of the response variable, e.g., to a logarithmic scale, corrects this situation. However, if there is a different pattern, then a transformation would not suffice to satisfy the homoschedasticity assumption.

Finally, the normality assumption is checked by testing if the residuals appear to be a random sample from a normal distribution. This is often done by a quantile-quantile normal probability plot, which is a plot of the standardized residuals against the theoretical quantiles of a population in fact having the normal distribution. In a quantile-quantile plot, a solid line passes through the first and third quantile. When the normality assumption is satisfied, then the points should follow roughly the straight line.

The checks of the requirements to perform ANOVA when MO-ACO was applied to unstructured instances of bQAP with $\xi \in \{0.75, 0.00, -0.75\}$ are plotted on Fig. A.1 for the configurations using local search methods and on Fig. A.2 for the configurations without local search. For comparison purposes, Figure A.3 shows the plots corresponding to configurations using local search when applied to structured instances.

The top row of every figure shows the Cook's distance plots. An unusual high value of the Cook's distance identifies an outlier and the numbers indicate the particular experiment (observation number) which caused the respective outlier. For every figure a few expected outliers are shown. The second row of the three figures shows the error independence plots. While in the first two figures the residuals are randomly distributed around zero with no discernible pattern, this is not the case in the third figure. In the third row, the Homoschedasticity plots show a random pattern around zero in the first figure, a less random pattern in the case of the second figure, and a evident megaphone-shaped pattern in the third figure. Various transformations of the data which corresponds to the third figure were not sufficient to eliminate the megaphone-shaped pattern. Finally, with regard to the quantile-quantile plots at the bottom row, in the first two figures the points follow roughly the straight line but the same is not true in the third figure. This fact indicates that the normality assumption is met for the data corresponding to unstructured instances and it is not met for the data of configurations using local search when applied to structured instances.

In summary, we conclude that the assumptions of ANOVA are in general met for the results obtained by MO-ACO for unstructured instances. In contrast, we conclude that the assumptions are not met for the configurations using local search methods when applied to structured instances. Nonetheless, in the latter case, the poor quality of the lower bound for structured instances (Fig. 6.1) would not allow us to obtain any conclusion from this data, even in the case that assumptions of ANOVA would have been satisfied.

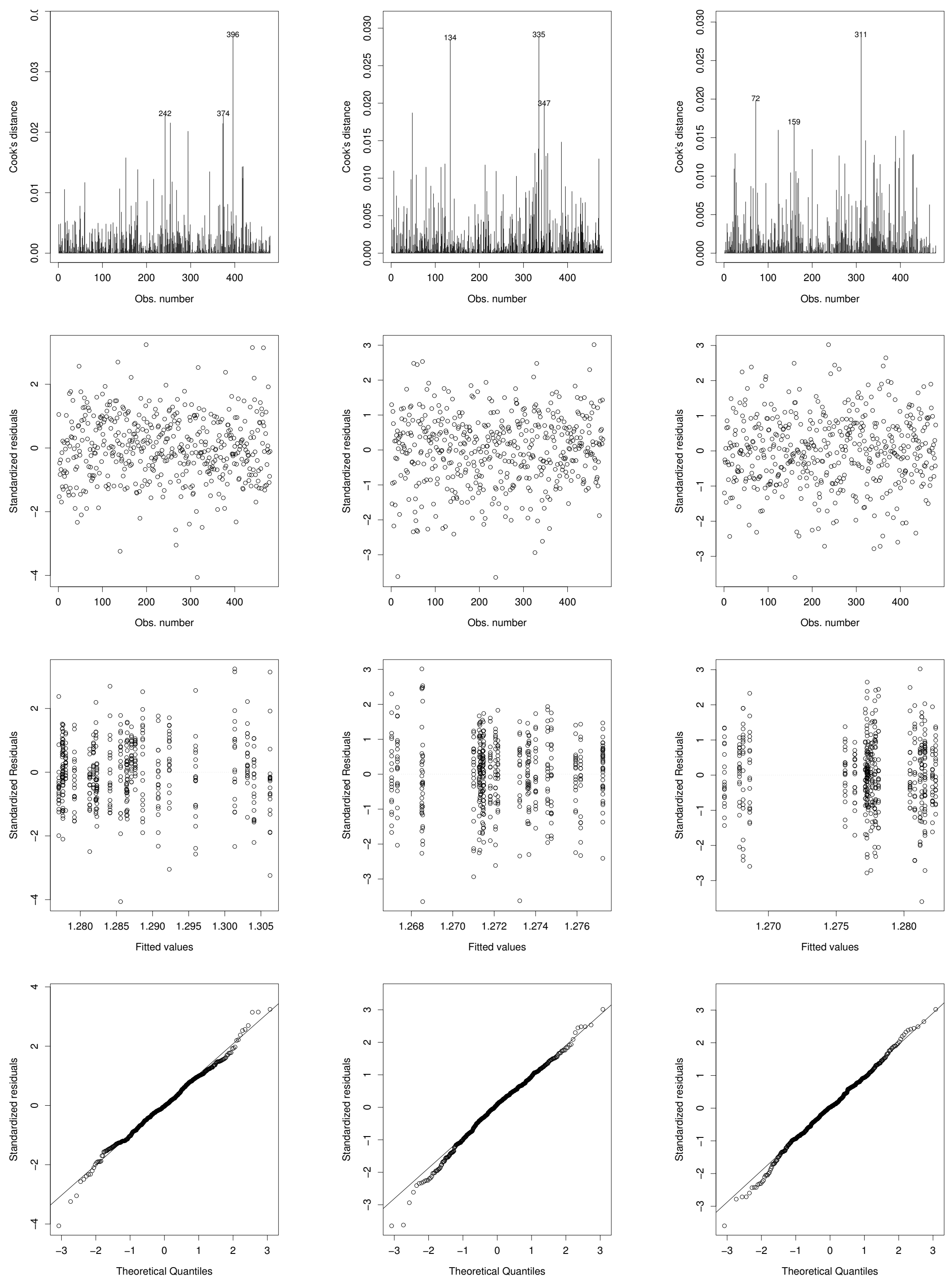

Figure A.1: Cook's distance (top row), Error Independence (2nd row), Homoschedasticity (3rd row) and Normality of residuals (bottom row) plots for *unstructured* instances with ξ of 0.75 (left column), 0.00 (middle column), -0.75 (right column), when using local search methods.

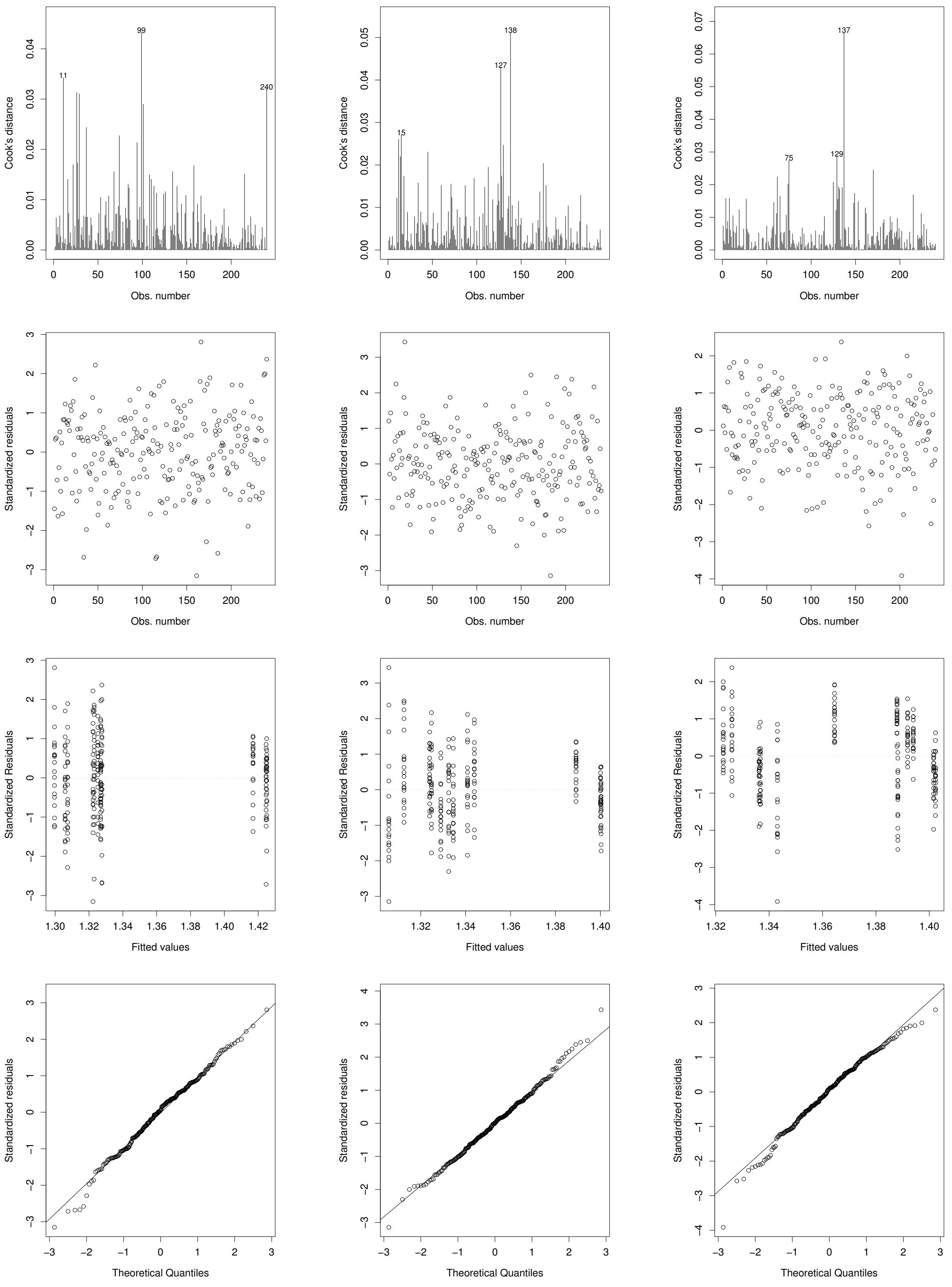

Figure A.2: Cook's distance (top row), Error Independence (2nd row), Homoschedasticity (3rd row) and Normality of residuals (bottom row) plots for *unstructured* instances with ξ of 0.75 (left column), 0.00 (middle column), -0.75 (right column), when *not* using local search methods.

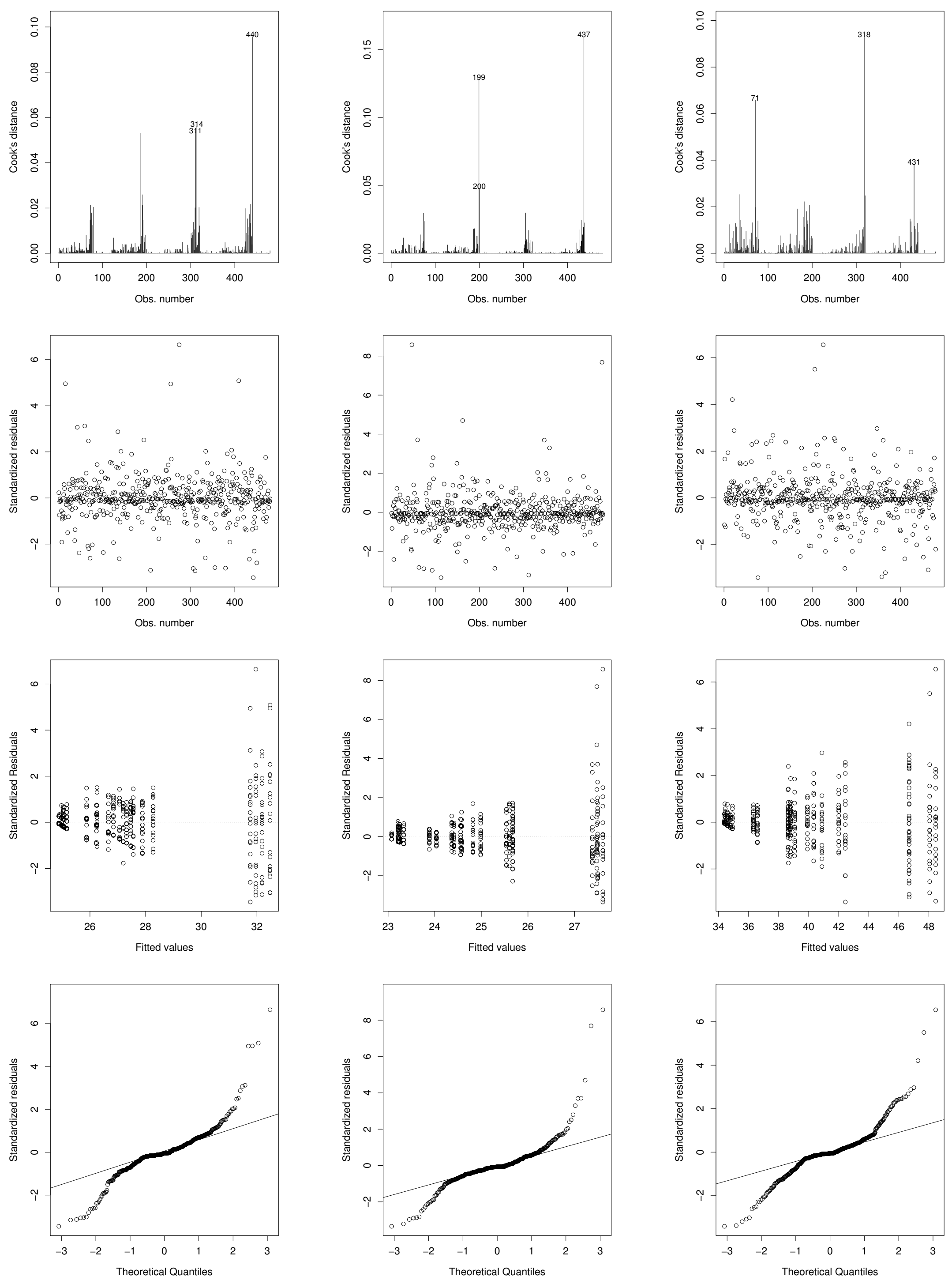

Figure A.3: Cook's distance (top row), Error Independence (2nd row), Homoschedasticity (3rd row) and Normality of residuals (bottom row) plots for *structured* instances with ξ of 0.75 (left column), 0.00 (middle column), -0.75 (right column), when using local search methods.

Bibliography

[1] R. Battiti and G. Tecchiolli. The reactive taboo search. *ORSA Journal on Computing*, 6(2):126–140, 1994.

[2] B. Bullnheimer, R. F. Hartl, and C. Strauss. An improved ant system algorithm for the vehicle routing problem. *Annals of Operations Research*, 89:319–328, 1999.

[3] R. E. Burkard, E. Çela, P. M. Pardalos, and L. S. Pitsoulis. The quadratic assignment problem. In P. M. Pardalos and D.-Z. Du, editors, *Handbook of Combinatorial Optimization*, volume 2, pages 241–338. Kluwer Academic Publishers, 1998.

[4] R. E. Burkard and J. Offerman. Entwurf von Schreibmaschinentastaturen mittels quadratischer Zuordnungsprobleme. *Zeitschrift für Operations Research*, 21:B121–B132, 1977.

[5] D. T. Connolly. An improved annealing scheme for the quadratic assignment problem. *European Journal of Operational Research*, 46(1):93–100, 1990.

[6] A. Dean and D. Voss. *Design and Analysis of Experiments*. Springer, 1999.

[7] J. W. Dickey and J. W. Hopkins. Campus building arrangement using TOPAZ. *Transportation Science*, 6:59–68, 1972.

[8] M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milano, Italy, 1992. (In Italian).

[9] M. Dorigo and G. Di Caro. The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 11–32. McGraw-Hill, London, 1999.

[10] M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant algorithms for distributed discrete optimization. *Artificial Life*, 5:137–172, 1999.

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

[12] M. Dorigo, V. Maniezzo, and A. Colorni. Positive feedback as a search strategy. Technical Report 91-016, Dipartimento di Elettronica e Informatica, Politecnico di Milano, Italy, 1991.

[13] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man and Cybernetics Part B: Cybernetics*, 26(1):29–41, 1996.

[14] M. Dorigo and T. Stützle. *Ant Colony Optimization.* MIT Press, Cambridge, MA, 2004.

[15] M. Ehrgott. *Multicriteria optimization*, volume 491 of *Lecture Notes in Economics and Mathematical Systems.* Springer-Verlag, Berlin, 2000.

[16] M. Ehrgott and X. Gandibleux. A survey and annoted bibliography of multiobjective combinatorial optimization. *OR Spectrum*, 22(4):425–460, 2000.

[17] H. A. Eiselt and G. Laporte. A combinatorial optimization problem arising in dartboard design. *Journal of the Operational Research Society*, 42:113–118, 1991.

[18] A. N. Elshafei. Hospital layout as a quadratic assignment problem. *Operations Research Quaterly*, 28:167–179, 1977.

[19] C. Fleurent and J. A. Ferland. Genetic hybrids for the quadratic assignment problem. In P. Pardalos and H. Wolkowicz, editors, *Quadratic assignment and related problems, DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, volume 16, pages 173–187. American Mathematical Society, 1994.

[20] L. M. Gambardella and M. Dorigo. Solving symmetric and asymmetric TSPs by ant colonies. In *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC'96)*, pages 622–627, Piscataway, USA, 1996. IEEE Press.

[21] L. M. Gambardella and M. Dorigo. An ant colony system hybridized with a new local search for the sequential ordering problem. *INFORMS Journal on Computing*, 12(3):237–255, 2000.

[22] L. M. Gambardella, É. D. Taillard, and G. Agazzi. MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 63–76. McGraw-Hill, London, 1999.

[23] L. M. Gambardella, É. D. Taillard, and M. Dorigo. Ant colonies for the quadratic assignment problem. *Journal of the Operational Research Society*, 50(2):167–176, 1999.

[24] A. M. Geoffrion and G. W. Graves. Scheduling parallel production lines with changeover costs: Practical applications of a quadratic assignment/LP approach. *Operations Research*, 24:595–610, 1976.

[25] P. C. Gilmore. Optimal and suboptimal algorithms for the quadratic assignment problem. *Journal of the SIAM*, 10:305–313, 1962.

[26] M. Gravel, W. L. Price, and C. Gagné. Scheduling continuous casting of aluminum using a multiple objective ant colony optimization metaheuristic. *European Journal of Operational Research*, 143(1):218–229, 2002.

[27] V. Grunert da Fonseca, C. M. Fonseca, and A. O. Hall. Inferential performance assessment of stochastic optimisers and the attainment function. In E. Zitzler, K. Deb, L. Thiele, C. A. C. Coello, and D. Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, volume 1993 of *Lecture Notes in Computer Science*, pages 213–225. Springer-Verlag, 2001.

[28] H. Hamacher, S. Nickel, and D. Tenfelde-Podehl. Facilities layout for social institutions. In *Operation Research Proceedings 2001 (OR2001)*, pages 229–236, Duisburg, September 2001.

[29] S. Iredi, D. Merkle, and M. Middendorf. Bi-criterion optimization with multi colony ant algorithms. In E. Zitzler, K. Deb, L. Thiele, C. C. Coello, and D. Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization, (EMO'01)*, volume 1993 of *Lecture Notes in Computer Science*, pages 359–372. Springer Verlag, Berlin, Germany, 2001.

[30] M. T. Jensen. Reducing the run-time complexity of multi-objective EAs: The NSGA-II and other algorithms. *IEEE Transactions on Evolutionary Computation*, 7(5):502–515, 2003.

[31] J. Knowles and D. Corne. Towards landscape analysis to inform the design of hybrid local search for the multiobjective quadratic assignment problem. In A. Abraham, J. Ruiz del Solar, and M. Koppen, editors, *Soft Computing Systems: Design, Management and Applications*, pages 271–279, Amsterdam, 2002. IOS Press. ISBN 1-58603-297-6.

[32] J. Knowles and D. Corne. Instance generators and test suites for the multiobjective quadratic assignment problem. In C. Fonseca, P. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, *Evolutionary Multi-Criterion*

Optimization, EMO 2003, volume 2632 of *Lecture Notes in Computer Science*, pages 295–310. Springer, 2003.

[33] J. Knowles and D. Corne. Bounded Pareto archiving: Theory and practice. In X. Gandibleux, M. Sevaux, K. Sörensen, and V. T'Kindt, editors, *Metaheuristics for Multiobjective Optimisation*, volume 535 of *Lecture Notes in Economics and Mathematical Systems*. Springer, January 2004. To appear.

[34] T. C. Koopmans and M. J. Beckmann. Asignment problems and the location of economic activities. *Econometrica*, 25:53–76, 1957.

[35] G. Laporte and H. Mercure. Balancing hydraulic turbine runners: A quadratic assignment problem. *European Journal of Operational Research*, 35:378–381, 1988.

[36] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. On the convergence and diversity-preservation properties of multi-objective evolutionary algorithms. Technical Report 108, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, May 2001.

[37] M. Laumanns, L. Thiele, E. Zitzler, and K. Deb. Archiving with guaranteed convergence and diversity in multi-objective optimization. In W. B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska, editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 439–447, New York, 9-13 July 2002. Morgan Kaufmann Publishers, San Francisco, CA 94104, USA.

[38] V. Maniezzo, A. Colorni, and M. Dorigo. The ant system applied to the quadratic assignment problem. Technical Report IRIDIA/94-28, IRIDIA, Université Libre de Bruxelles, Belgium, 1994.

[39] C. E. Mariano and E. Morales. MOAQ an ant-Q algorithm for multiple objective optimization problems. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 894–901, Orlando, Florida, USA, 13-17 July 1999. Morgan Kaufmann Publishers, San Francisco, CA 94104, USA.

[40] P. Merz and B. Freisleben. Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. *IEEE Transactions on Evolutionary Computation*, 4(4):337–352, 2000.

[41] C. H. Papadimitriou and M. Yannakakis. On the approximability of trade-offs and optimal access of Web sources. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science: 12–14 November, 2000, Redondo Beach, California*, pages 86–92. IEEE Computer Society Press, 2000.

[42] L. Paquete, M. Chiarandini, and T. Stützle. Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study. In X. Gandibleux, M. Sevaux, K. Sörensen, and V. T'Kindt, editors, *Metaheuristics for Multiobjective Optimisation*, volume 535 of *Lecture Notes in Economics and Mathematical Systems*. Springer Verlag, 2004.

[43] L. Paquete and T. Stützle. A study of local search algorithms for the biobjective QAP with correlated flow matrices. *European Journal of Operational Research*, 2004. Tam.

[44] R Development Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2004.

[45] S. Sahni and T. Gonzalez. P-complete approximation problems. *Journal of the ACM*, 23:555–565, 1976.

[46] L. Steinberg. The backboard wiring problem: a placement algorithm. *SIAM Review*, 3:37–50, 1961.

[47] R. E. Steuer. *Multiple Criteria Optimization: Theory, Computation and Application*. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, New York, NY, 1986.

[48] T. Stützle. *Local Search Algorithms for Combinatorial Problems: Analysis, Improvements, and New Applications*, volume 220 of *DISKI*. Infix, Sankt Augustin, Germany, 1999.

[49] T. Stützle and H. H. Hoos. The $\mathcal{MAX}$–$\mathcal{MIN}$ ant system and local search for the traveling salesman problem. In T. Bäck, Z. Michalewicz, and X. Yao, editors, *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation (ICEC'97)*, pages 309–314. IEEE Press, Piscataway, NJ, 1997.

[50] T. Stützle and H. H. Hoos. $\mathcal{MAX}$–$\mathcal{MIN}$ ant system and local search for combinatorial optimization problems. In S. Voss, S. Martello, I. Osman, and C. Roucairol, editors, *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, pages 137–154. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.

[51] T. Stützle and H. H. Hoos. $\mathcal{MAX}$-$\mathcal{MIN}$ ant system. *Future Generation Computer Systems*, 16(8):889–914, 2000.

[52] É. D. Taillard. Robust taboo search for the quadratic assingnment problem. *Parallel Computing*, 17:443–455, 1991.

[53] É. D. Taillard. A comparison of iterative searches for the quadratic assignment problem. *Location Science*, 3:87–105, 1995.

[54] É. D. Taillard. Fant: Fast ant system. Technical Report IDSIA-46-98, IDSIA, Lugano, Switzerland, 1998.

[55] É. D. Taillard and L. M. Gambardella. Adaptative memories for the quadratic assignment problem. Technical Report IDSIA-87-97, IDSIA, Lugano, Switzerland, 1997.

[56] M. Visée, J. Teghem, M. Pirlot, and E. L. Ulungu. Two-phases method and branch and bound procedures to solve the bi-objective knapsack problem. *Journal of Global Optimization*, 12:139–155, 1998.

[57] T. Vollmann and E. Buffa. The facilities layout problem in perspective. *Management Sciencie*, 12(10):450–468, 1966.

[58] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7:117–132, 2003.