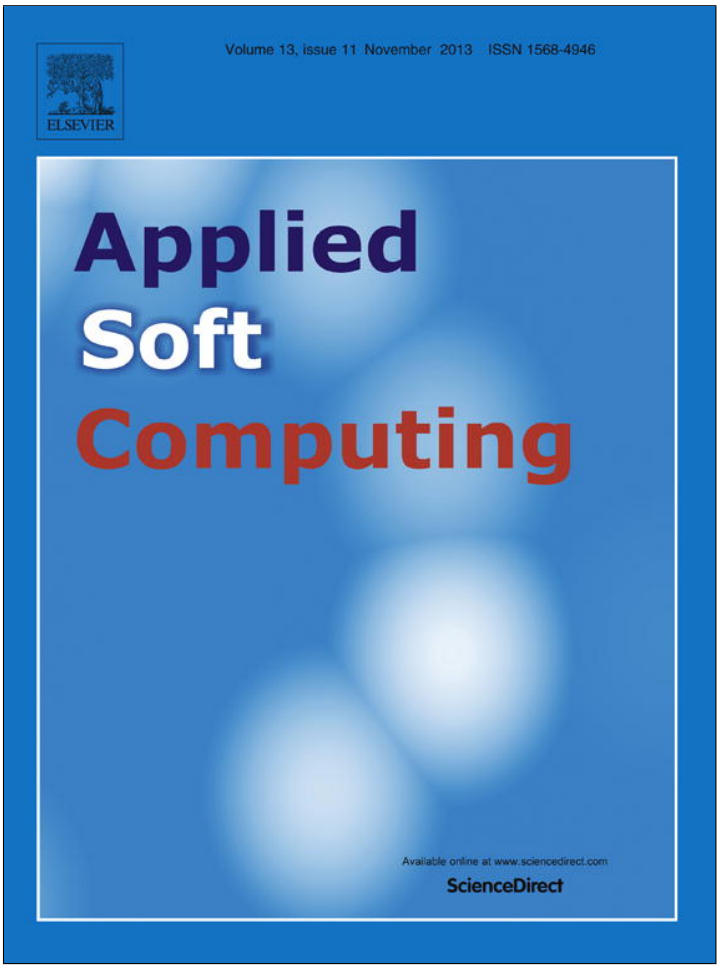

Applied Soft Computing 13 (2013) 4370–4382

Contents lists available at ScienceDirect

Applied Soft Computing

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

A comparative study of Multi-Objective Ant Colony Optimization algorithms for the Time and Space Assembly Line Balancing Problem

Juan Rada-Vilela[a,*], Manuel Chica[b], Óscar Cordón[b,c], Sergio Damas[b]

[a] Victoria University of Wellington, 6140 Wellington, New Zealand
[b] European Centre for Soft Computing, 33600 Mieres, Asturias, Spain
[c] DECSAI and CITIC-UGR, University of Granada, 18071 Granada, Spain

ARTICLE INFO

Article history:
Received 19 June 2012
Received in revised form 6 June 2013
Accepted 22 June 2013
Available online 3 July 2013

Keywords:
Ant Colony Optimization
Multi-Objective Optimization
Time and Space Assembly Line Balancing Problem
Automotive industry

ABSTRACT

Assembly lines for mass manufacturing incrementally build production items by performing tasks on them while flowing between workstations. The configuration of an assembly line consists of assigning tasks to different workstations in order to optimize its operation subject to certain constraints such as the precedence relationships between the tasks. The operation of an assembly line can be optimized by minimizing two conflicting objectives, namely the number of workstations and the physical area these require. This configuration problem is an instance of the TSALBP, which is commonly found in the automotive industry. It is a hard combinatorial optimization problem to which finding the optimum solution might be infeasible or even impossible, but finding a good solution is still of great value to managers configuring the line. We adapt eight different Multi-Objective Ant Colony Optimization (MOACO) algorithms and compare their performance on ten well-known problem instances to solve such a complex problem. Experiments under different modalities show that the commonly used heuristic functions deteriorate the performance of the algorithms in time-limited scenarios due to the added computational cost. Moreover, even neglecting such a cost, the algorithms achieve a better performance without such heuristic functions. The algorithms are ranked according to three multi-objective indicators and the differences between the top-4 are further reviewed using statistical significance tests. Additionally, these four best performing MOACO algorithms are favourably compared with the Infeasibility Driven Evolutionary Algorithm (IDEA) designed specifically for industrial optimization problems.

1. Introduction

Assembly lines consist of workstations where products are built after a number of tasks are performed. These tasks are distributed between the workstations according to time and space requirements as well as the respective order of precedence between the tasks. Thus, products are built incrementally by flowing from workstation to workstation. These flow-oriented production systems are commonly found in the mass-production industry. However, its configuration is a very complex combinatorial optimization problem known as the Assembly Line Balancing Problem (ALBP) [1].

The first family of problems to model such a configuration was the Simple Assembly Line Balancing Problem (SALBP) [2,1], which is a general class of bin-packing problems with additional precedence constraints. In this family, the tasks must be distributed between the workstations with the objective of minimizing the inefficiency of the line (or its total downtime) subject to the constraints imposed on the tasks and workstations. However, the configurations modeled in this family are too general and limit the representation of cases such as those in the automotive industry where space constraints must also be considered.

The Time and Space Assembly Line Balancing Problem (TSALBP) is an extension of SALBP that incorporates the constraints of time and space into the model [3]. Thus, TSALBP provides a more accurate representation of real-world problems where three optimization objectives are to be minimized: the number of workstations, the physical area these occupy, and the time required to finish their respective tasks. From this family, different problems can be derived according to the combination of these objectives.

One such problems is the TSALBP-m/A[1] in which the objectives to minimize are the number of workstations and their required physical area according to a known fixed cycle time [4–6]. Thus,

* Corresponding author.
E-mail addresses: juan.rada-vilela@ecs.vuw.ac.nz (J. Rada-Vilela), manuel.chica@softcomputing.es (M. Chica), ocordon@decsai.ugr.es (Ó. Cordón), sergio.damas@softcomputing.es (S. Damas).

[1] Originally, this TSALBP variant is referred as TSALBP-1/3 [3]. This new notation is introduced in this work for a better understanding.

1568-4946/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.asoc.2013.06.014

instead of three objectives, TSALBP-m/A is a bi-objective optimization problem with an additional constraint of time. The number of workstations and their required physical area are conflicting objectives because the space requirements of the workstations are determined by the tasks assigned therein. Thus, assigning all tasks between a few workstations will increase the space requirements, whereas assigning them between many workstations will reduce them. The complexity of this problem makes it practically impossible in most cases to perform an exhaustive search to find the optimum solution. Thus, most of these problems are preferably approached with metaheuristics in order to find good solutions within a reasonable amount of time.

Ant Colony Optimization (ACO) is a metaheuristic that models optimization problems as graphs which artificial ants explore to build potential solutions to the problem at hand [7]. Once each ant has built a candidate solution, they lay pheromone trails over the search space to encourage others to further explore the surroundings of the best solutions found. The multi-objective nature of the TSALBP has encouraged the use of multi-objective ACO (MOACO) algorithms to tackle these problems [8,9] as well different formulations of the ALBP [10,11].

The overall goal of this article is to provide a comparative study of state-of-the-art MOACO algorithms and adapt them to tackle the TSALBP-m/A. Specifically, we will adapt, evaluate, compare and rank the performance of eight different MOACO algorithms on ten well-known TSALBP-m/A instances using several multi-objective performance indicators. Particularly, we are interested in investigating the underlying reasons for such a detriment in performance when utilizing the heuristic information in our framework. In addition, the performance of the MOACO algorithms will be compared with the multi-objective Infeasibility Driven Evolutionary Algorithm (IDEA) which is explicitly designed for industrial constrained optimization problems [12].

An important component of MOACO algorithms is the use of heuristic information to improve the quality of the solutions found. If the heuristic information provides additional insights about the problem at hand, it can only be expected that algorithms using such information will provide much better solutions than algorithms ignoring it. However, a previous work [4] has shown that significantly better results are obtained in the TSALBP-m/A when such information is not utilized.

The remainder of this article is structured as follows. Section 2 presents a background on TSALBP, ACO, adapting general MOACO algorithms to the TSALBP-m/A, and some multi-objective performance indicators. Section 3 describes the operation and specific adaptation of each of the MOACO algorithms considered to the TSALBP-m/A. Section 4 describes the experimental design for our research goals. Section 5 presents the results and our respective discussions. Finally, Section 6 presents the conclusions and suggestions for future research on this topic.

2. Background

2.1. Time and Space Assembly Line Balancing Problem

The Time and Space Assembly Line Balancing Problem (TSALBP) models the configuration of an assembly line whose tasks required to build a product must be distributed between a number of workstations. Each of these tasks have time and space requirements as well as precedence relationships with other tasks. For example, a TSALBP instance problem is presented in Fig. 1 as an acyclic graph where nodes represent tasks with time and space requirements and the directed edges represent the precedence constraints.

The TSALBP is modeled as a set V of n tasks which are distributed into workstations. Each workstation k has a subset $S_k \in V$ of tasks

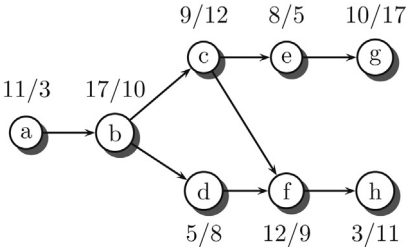

Fig. 1. Example of a TSALBP precedence graph.

to be performed within a cycle time c before passing the product to the next workstation. The workload time $t(S_k)$ and required area $a(S_k)$ are respectively defined as the sum of the operation times and required areas of the tasks in S_k. Thus, the TSALBP is formulated as the distribution of n tasks, each with positive time and space requirements, subject to the following constraints:

- Each task j is assigned to one workstation k only, i.e.,

$$\sum_{k=E_j}^{L_j} x_{jk} = 1, \quad 1 \le j \le n \tag{1}$$

  where $x_{jk} \in \{0, 1\}$ determines whether task j is assigned to workstation k, E_j and L_j are the earliest and latest workstations to which task j may be assigned (respectively), and n is the number of tasks.
- All tasks are distributed into m workstations at most, i.e.,

$$\sum_{k=1}^{M} \max_{j=1,2,\ldots,n} x_{jk} \le m \tag{2}$$

  where M is the maximum the number of workstations.
- The workload time for any workstation is at most the cycle time c, i.e.,

$$\sum_{j=1}^{n} t_j x_{jk} \le c, \quad k = 1, 2, \ldots, M \tag{3}$$

  where t_j is the operation time of task j.
- The required area for any workstation is at most A, i.e.,

$$\sum_{j=1}^{n} a_j x_{jk} \le A, \quad k = 1, 2, \ldots, M \tag{4}$$

  where a_j is the area required for task j.
- The distribution of tasks satisfy the precedence relationships between them, i.e.,

$$\sum_{k=E_i}^{L_i} k x_{ij} \le \sum_{k=E_j}^{L_j} k x_{jk}, \quad j = 1, 2, \ldots n; \forall i \in P_j \tag{5}$$

  where P_j is the set of immediate predecessor tasks that must be completed before task j.

The TSALBP family groups a total of 8 optimization problems according to the combinations of the different objectives to optimize, namely (1) the number of workstations m, (2) the cycle time c, and (3) the maximum physical area A occupied by any given workstation. As such, the TSALBP-m/A is a problem in which the objectives to optimize are m and A while satisfying the aforementioned constraints. Mathematically, these objectives are:

$$\text{Minimize} \quad f^s(x) = m = \sum_{k=1}^{M} \max_{j=1,2,\ldots,n} x_{jk} \tag{6}$$

$$f^a(x) = A = \max_{k=1,2,\ldots,M} \sum_{j=1}^{n} a_j x_{jk} \tag{7}$$

2.2. Ant Colony Optimization

Ant Colony Optimization (ACO) is a metaheuristic designed by Dorigo and some other contributors [7] with inspiration on the behaviour of ants for finding the shortest path from their nest to the food source. Its design consists of a number of ants that collectively explore the solution search space and lay pheromone trails in order to aid other ants into finding better solutions. ACO works by having ants deposit pheromones over the trails used to find the solutions, and the amount of pheromones is relative to the quality of the solutions. As such, pheromone trails that lead to better solutions will be stronger than those leading to worse ones, and ants will be more likely attracted to follow the trails with more pheromones.

ACO is an iterative algorithm in which the problem is represented by a connected graph and its solutions by an ordered sequence of connected nodes. At each iteration, ants explore the solution search space by probabilistically moving from one node to another. Such a decision is determined by the *transition rule*, which considers the amount of pheromone deposited between the pair of connected nodes and the heuristic information expressing the preference between them. Once the ants have built their complete solutions, i.e. each ant has explore all the nodes, their objective values are utilized to determine the amount of pheromones to deposit along the trails. Additionally, at the end of each iteration, the amount of pheromone over the trails is reduced by a factor coined *evaporation rate* in order to prevent stagnation and encourage the exploration of new solutions.

The first ACO algorithm proposed in the literature was the (AS) [13], in which the pheromone τ_{ij} deposited between nodes i and j is determined as follows,

$$\tau_{ij} = (1-\rho)\cdot\tau_{ij} + \sum_{k=1}^{\Gamma} \Delta\tau_{ij}^k \tag{8}$$

where ρ is the evaporation rate, Γ is the number of ants, and $\Delta\tau_{ij}^k$ is the amount of pheromone deposited between i and j by ant k according to the quality of its solution. The transition rule is determined by the probability of ant k to visit node j from node i as follows,

$$p_{ij}^k = \frac{\tau_{ij}^\alpha \cdot \eta_{ij}^\beta}{\sum_{u\in\Omega} \tau_{iu}^\alpha \cdot \eta_{iu}^\beta} \tag{9}$$

where η_{ij} is a heuristic function that quantifies the preference of ant k at node i to visit j, α and β are factors that determine the influence of the pheromone and the heuristic function (respectively), and Ω is the set of nodes available from i.

2.3. Multi-Objective Ant Colony Optimization and the Time and Space Assembly Line Balancing Problem

The TSALBP-m/A is a bi-objective optimization problem in which the number of workstations and the maximum physical area any of these occupy must be minimized. However, both objectives are in conflict as the minimization of the number of workstations implies that each will have more tasks assigned and hence increase their physical area. Conversely, minimizing the maximum physical area implies that less tasks need to be assigned to the workstations and hence it will increase the number of workstations to be able to assign all the tasks. Therefore, solutions to this problem must balance a tradeoff between both objectives.

The multi-objective nature of the TSALBP-m/A, its conflicting objectives, and its natural representation by means of acyclic directed graphs, makes it appropriate to utilize MOACO algorithms to find good solutions to this type of problem. Particularly, we are interested in the Pareto-based MOACO algorithms as these find a set of non-dominated solutions (known as a Pareto front) which provide different tradeoffs between the objectives [14].

Many MOACO algorithms have been proposed in the literature to address a wide range of optimization problems. However, in order to address the TSALBP-m/A, we need to adapt them such that they have the following characteristics: a workstation-oriented scheme, a heuristic function to each objective, and a Pareto archive to store the solutions. Each of these characteristics is detailed in the following sections.

2.3.1. Station-oriented scheme

In the station-oriented scheme [1,3], the pheromone matrix τ_{ij} contains the pheromone trail expressing the preferability of assigning task j to workstation i. At each iteration, ant k starts with an opened workstation s_1^k to which tasks are assigned via the transition rule. After each assignment, the ant probabilistically decides to close the current station i and open a new one based on the filling rate $\sigma(s_i^k)$ and a threshold θ_k according to (10),

$$p(s_{i+1}^k) = \begin{cases} 1, & \text{if } [\sigma(s_i^k) \geq \theta_k] \wedge [\sigma(s_i^k) > r] \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

where $r \in [0, 1]$ is a random value sampled from a uniform distribution, and $\sigma(s_i)$ is the filling rate of the current workstation computed as the ratio between the time accumulated by every task t and the cycle time c according to (11),

$$\sigma(s_i) = \frac{\sum_{j\in s_i} t_j}{c} \tag{11}$$

This probabilistic method of opening workstations encourages diversity and hence provides a better the Pareto front [6]. Since the threshold θ_k of each ant determines the probability for the ant to completely fill its opened workstation, a low threshold will open more workstations and hence have fewer tasks in each of them, whereas a high threshold will open less workstations and hence more tasks in each of them. Therefore, diversity is further increased by assigning different thresholds to ants.

2.3.2. Heuristic function

The heuristic function provides guidance about the preferability to assign task j to the currently open workstation i. This information is computed for task j based on its required time t_j and its corresponding required area a_j [3]. Specifically, the heuristic function is computed for each objective according to (12) and (13), respectively,

$$\eta_{ij}^s = \frac{t_j}{c} \cdot \frac{\left|F_j\right| + 1}{\max_{u\in\Omega} \left|F_u\right| + 1} \tag{12}$$

$$\eta_{ij}^a = \frac{a_j}{A_\Sigma} \cdot \frac{\left|F_j\right| + 1}{\max_{u\in\Omega} \left|F_u\right| + 1} \tag{13}$$

where A_Σ is the total area required by all tasks in the problem, $\left|F_j\right|$ is the number of tasks available from j, and $\max_{u\in\Omega} \left|F_u\right|$ is the maximum number of tasks available from any task following j. In the case of MOACO algorithms whose design involves a single heuristic function, we merge both of them into one as $\eta_{ij} = \eta_{ij}^s \cdot \eta_{ij}^a$.

2.3.3. Pareto archive

MOACO algorithms usually have a Pareto archive in which only non-dominated solutions are stored. After each iteration, all the solutions found by the ants are compared against those stored in the archive. If a non-dominated solution is found, then it will be added to the archive and the newly dominated ones will be removed and discarded. As such, the size of the Pareto archive is subject to change after each iteration. This might pose a computational challenge when the range of the objective values is too large or these are represented with a large number of decimals. In both cases, strategies to truncate the size of the archive may be needed. Nonetheless, the objective values of the solutions to the problem instances which we will focus on are all integers within a specific range, and hence no truncation mechanism is required.

2.4. Performance indicators

In single-objective optimization, the performance of two algorithms can be compared directly by just considering the objective values of the best solutions found. As such, in a minimization problem, the best-performing algorithm will be the one which finds the solutions with the lowest objective values. However, in multi-objective optimization, there is no such thing as a unary indicator to absolutely distinguish the quality between any two non-dominated sets of solutions. Therefore, it is necessary to define a criterion to differentiate the performance of two algorithms with respect to the quality of their Pareto fronts. Given that MOACO algorithms are intrinsically approximation methods, hereafter we refer to their sets of non-dominated solutions as their *approximate* Pareto fronts, and to the optimal set solving the problem as the *true* Pareto front. These terms are utilized to define the following performance indicators proposed in the literature.

2.4.1. The hyper-volume indicator

The hyper-volume indicator I_H [15] measures the coverage of an approximate Pareto front with respect to the true one. As such, an algorithm is better than another if its approximate Pareto front has a greater coverage. For each algorithm, this indicator is computed as follows,

$$I_H = \frac{HV(P)}{HV(P^*)} \tag{14}$$

where $HV(P)$ and $HV(P^*)$ are the hyper-volumes of the approximate and the true Pareto fronts, respectively. Thus, the higher the value of I_H, the closer the approximate Pareto front is to the true one, and when $I_H = 1$ both Pareto fronts are equal.

The hyper-volume of a Pareto front is computed with respect to a reference point z_{ref} defined according to the problem. It is necessary that such a reference is dominated by all the solutions in the front. Thus, the reference point for each objective is set as the minimum value possible in maximization problems, whereas to the maximum value in minimization ones. However, these values must be set within reasonable distance as if set too far it might compromise the accuracy of the indicator [16]. Fig. 2 shows the hyper-volume indicator in gray of a Pareto front with respect to a reference point in both minimization and maximization problems.

2.4.2. The epsilon indicator

The ϵ-indicator [17] measures the factor by which an approximate Pareto front is worse than the true one with respect to all the objectives. Specifically, $I_\epsilon(P, P^*)$ is the minimum factor by which a solution in P is worse than any solution in P^*. It is computed as follows:

$$I_\epsilon(P, P^*) = \max_{\mathbf{z}^2 \in P^*} \min_{\mathbf{z}^1 \in P} \max_{1 \le i \le n} \frac{z_i^1}{z_i^2} \tag{15}$$

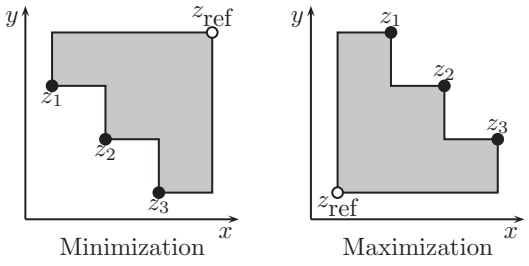

Fig. 2. Hyper-volume indicator.

where $\mathbf{z}^1 = (z_1^1, \ldots, z_n^1)$ and $\mathbf{z}^2 = (z_1^2, \ldots, z_n^2)$ are the vectors of solutions in sets P and P^*, respectively.

2.4.3. The coverage indicator

The coverage indicator I_C [15] measures the quality of an approximate Pareto front P with respect to another front Q by the ratio between the number of solutions in P that dominate a solution in Q. Specifically, this indicator is computed as follows,

$$I_C(P, Q) = \frac{|\{q \in Q; \exists p \in P : p \prec q\}|}{|Q|} \tag{16}$$

where $p \prec q$ indicates that solution $p \in P$ dominates solution $q \in Q$ in a minimization problem. As such, when $I_C(P, Q) = 1$, all the solutions in Q are dominated by the solutions in P. Conversely, when $I_C(P, Q) = 0$ none of the solutions in Q are dominated by a single solution in P. Notice that this relation does not need to be complementary, that is, $I_C(P, Q) = 1 - I_C(Q, P)$ does not necessarily hold.

An indicator derived from I_C is the thresholded coverage indicator I_C^θ [18], which is useful for comparing two algorithms with multiple approximate Pareto fronts obtained in different repetitions. The thresholded coverage indicator is computed as follows,

$$I_C^\theta(P_i, Q_i) = \begin{cases} 1 & \text{if } I_C(P_i, Q_i) > \theta \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

where $\theta \in (0.5, 1.0]$ is the threshold value. Thus, if $I_C^\theta(P_i, Q_i) = 1$ the Pareto front P_i dominates Q_i at repetition i, and the average of the I_C^θ computed from every repetition provides an indicator of quality between the two algorithms.

3. Multi-Objective Ant Colony Optimization Algorithms

MOACO algorithms are especially designed to tackle multi-objective optimization problems in which an important number of constraints should be satisfied. Many different MOACO designs have been proposed [9] and they have demonstrated an outstanding performance. Among them, we are interested in all Pareto-based approaches that handle multiple objectives by utilizing one or more pheromone matrices and heuristic functions. Specifically, we adapt the following MOACO algorithms to tackle the TSALBP-m/A:

- Bi-Criterion Optimization with a Single Colony,
- Bi-Criterion Optimization with Multiple Colonies, updates by origin,
- Bi-Criterion Optimization with Multiple Colonies, updates by region,
- Competing Ant Colonies (COMPETants),
- Multiple Ant Colony System (MACS),
- Multi-Objective Ant-Q (MOAQ),
- Multi-Objective Network Ant Colony Optimization (MONACO), and
- Pareto Ant Colony Optimization (PACO).

Table 1
Taxonomy of the MOACO algorithms.

	Heuristic functions	
	Single	Multiple
Pheromone matrices		
Single		MACS
		MOAQ
Multiple	MONACO	All Bi-Criterion
	PACO	COMPETants

These algorithms may be classified according to the number of pheromone matrices and heuristic functions as shown in Table 1 [8]. The following subsections are devoted to briefly introduce the main components of each algorithm as well as the customization we have made to apply them to the TSALBP-M/A

3.1. Bi-Criterion Optimization with a Single Colony

The Bi-Criterion Optimization with a Single Colony [19] utilizes one pheromone matrix and one heuristic function for each objective, thus having two of each to tackle the TSALBP-m/A. The transition rule in this algorithm is determined by (18) as follows,

$$p_{ij} = \frac{[\tau_{ij}^{s}]^{\lambda\alpha}[\tau_{ij}^{a}]^{(1-\lambda)\alpha}[\eta_{ij}^{s}]^{\lambda\beta}[\eta_{ij}^{a}]^{(1-\lambda)\beta}}{\sum_{u\in\Omega}[\tau_{iu}^{s}]^{\lambda\alpha}[\tau_{iu}^{a}]^{(1-\lambda)\alpha}[\eta_{iu}^{s}]^{\lambda\beta}[\eta_{iu}^{a}]^{(1-\lambda)\beta}} \tag{18}$$

where $\lambda = (k-1)/(\Gamma-1)$ is computed according to the number of ants Γ and the index of ant $k \in [1, \Gamma]$. Thus, ants search different regions of the Pareto front by utilizing different weights that determine the relative importance of the optimization criteria.

After each ant k has built its solution S_k, pheromone trails of both matrices are evaporated by a factor of ρ, and only those ants with non-dominated solutions in the current iteration are allowed to update the pheromone trails. Originally, such an update was performed according to $\tau_{ij} = \tau_{ij} + 1/\Gamma_{\star}$, where $\Gamma_{\star}$ is the number of ants allowed to update. However, this kind of update entails that the two pheromone matrices are updated with the same amount, thus resulting in two equivalent matrices. In order to avoid such a redundancy, we utilize the inverse cost of their respective objective functions as proposed in [8]. Hence, the pheromone update rule is instead given by (19).

$$\tau_{ij}^{o} = \tau_{ij}^{o} + \frac{1}{f^{o}(S)} \tag{19}$$

3.2. Bi-Criterion Optimization with Multiple Colonies and Update by Origin

The Bi-Criterion Optimization with Multiple Colonies and Update by Origin [19] utilizes one pheromone matrix and one heuristic function for each objective, thus having two of each to tackle the TSALBP-m/A. The transition rule of this algorithm is the same as the single colony approach (18), but uses instead overlapping λ-values amongst colonies.

The pheromone updates by origin are performed by all ants with a globally non-dominated solution using the same rule as the single colony approach (19). As such, this method imposes a strong selection pressure given that ants have to find a non-dominated solution not only locally but globally in order to be allowed to update the pheromone matrices. However, it also favours diversity as it encourages colonies to search in different regions with less dense areas where ants will be more likely to update the pheromone matrices.

3.3. Bi-Criterion Optimization with Multiple Colonies and Update by Region

The Bi-Criterion Optimization with Multiple Colonies and Update by Region [19] is similar to the previous algorithm except for the pheromone update method. In this case, the pheromone update is performed by region as ant colonies are explicitly guided towards different search regions of the Pareto front. Specifically, the list of globally non-dominated solutions in the current iteration is sorted according to either of the objectives, and then it is split up into p equally-sized parts matching the number of colonies. For each part i, only those ants that have found a solution update the pheromone matrices of colony i according to (19).

3.4. Competing Ant Colonies

COMPETants [20] was originally proposed to deal with bi-objective transportation problems. It consists of two ant colonies of variable size with different priority rules over the objectives to optimize. These priority rules are determined according to the heuristic information. In the case of the TSALBP-m/A, the first colony utilizes the heuristic information related to time, and the other colony utilizes that related to the area.

In this algorithm, each ant uses the pheromone and heuristic information from the colony c it belongs to. Thus, the transition rule is given by (20).

$$p(j) = \begin{cases} \dfrac{[\tau_{ij}^{c}]^{\alpha}[\eta_{ij}^{c}]^{\beta}}{\sum_{u\in\Omega}[\tau_{iu}^{c}]^{\alpha}[\eta_{iu}^{c}]^{\beta}} & \text{if } j \in \Omega \\ 0, & \text{otherwise} \end{cases} \tag{20}$$

After each iteration, ants in each colony have a probability $\psi_{\text{own}}/(4\psi_{\text{for}} + \psi_{\text{own}})$ to become *spies*, where ψ_{own} and ψ_{for} are the product of the number of workstations and the maximum area $(m \times A)$ of the best solution found in the own and foreign colony, respectively. As such, the spy's transition rule is changed to (21).

$$p(j) = \begin{cases} \dfrac{\left[0.5\tau_{ij}^{\text{own}} + 0.5\tau_{ij}^{\text{for}}\right]^{\alpha}\left[\eta_{ij}^{\text{own}}\right]^{\beta}}{\sum_{u\in\Omega}\left[0.5\tau_{iu}^{\text{own}} + 0.5\tau_{iu}^{\text{for}}\right]^{\alpha}\left[\eta_{iu}^{\text{own}}\right]^{\beta}}, & \text{if } j \in \Omega \\ 0, & \text{otherwise} \end{cases} \tag{21}$$

Once ants have built their solutions, the pheromone of both colonies evaporates by ρ. Then, only the best Λ-ants of each colony are allowed to update the pheromone trails of their respective solutions according to (22),

$$\tau_{ij}^{o} = \tau_{ij}^{o} + 1 - \frac{k_{\Lambda} - 1}{\Lambda}, \quad \forall i, j \in S_{k_{\Lambda}} \tag{22}$$

where Λ is equal to 6.25% of the total size of colony c ($\Lambda = \lceil 0.0625 \cdot \Gamma^{c} \rceil$), and $k_{\Lambda} \in [1, \Lambda]$ is the position of the ant in the ranked list.

Finally, the size of the colonies is adapted in such a way that the colony with the best average μ^{c} of solutions has a higher probability of receiving more ants from the other colony. For the TSALBP-m/A, the best average of solutions is defined as the average product of the number of workstations and maximum area. Thus, ants from both colonies are assigned to the first colony with probability $\mu^{\text{for}}/(\mu^{\text{own}} + \mu^{\text{for}})$, and all the remaining ants are assigned to the second colony, ensuring that at least one ant is present in each of them.

3.5. Multiple Ant Colony System

MACS [21] is an algorithm based on the Ant Colony System [22], and it uses a single pheromone matrix and two heuristic functions

related to the objectives to optimize. The transition rule to move from node i to j is determined as follows,

$$j = \begin{cases} \underset{j\in\Omega}{\arg\max}\{\tau_{ij}[\eta^s_{ij}]^{\lambda\beta}[\eta^a_{ij}]^{(1-\lambda)\beta}\}, & \text{if } q \le q_0 \\ z, & \text{otherwise} \end{cases} \quad (23)$$

where $q_0 \in [0, 1]$ is a predefined constant, $q \sim U(0, 1)$ is a random number sampled from a uniform distribution, z is a node selected according to the probability distribution in (24), and λ is computed as the ratio of the ants' position index $k \in [1, \Gamma]$ with regards to the size of the colony according to $\lambda = (k-1)/(\Gamma-1)$.

The transition rule utilized by this algorithm is known as the pseudo-random-proportional rule, and it is defined according to (24).

$$p(j) = \begin{cases} \dfrac{\tau_{ij}[\eta^s_{ij}]^{\lambda\beta}[\eta^a_{ij}]^{(1-\lambda)\beta}}{\sum_{u\in\Omega}\tau_{iu}[\eta^s_{iu}]^{\lambda\beta}[\eta^a_{iu}]^{(1-\lambda)\beta}}, & \text{if } j \in \Omega \\ 0, & \text{otherwise} \end{cases} \quad (24)$$

After the transition rule takes place, each ant updates its pheromone trail as follows,

$$\tau_{ij} = (1-\rho)\,\tau_{ij} + \rho\tau_0 \quad (25)$$

where τ_0 is computed initially as the inverse product of the average number of workstations and the maximum area of solutions found heuristically according to (26).

$$\tau_0 = \frac{1}{\hat{f}^s \cdot \hat{f}^a} \quad (26)$$

Once all ants have built their complete solutions, these are compared to those in the Pareto archive and the non-dominated solutions are added while removing the newly dominated ones. Afterwards, τ'_0 is computed from the Pareto archive according to (26). Finally, if $\tau'_0 > \tau_0$, the pheromone trails are reinitialized to τ'_0, otherwise, the pheromone trails associated to each solution S in the Pareto archive are updated according to (27).

$$\tau_{ij} = (1-\rho)\tau_{ij} + \frac{\rho}{f^s(S)\cdot f^a(S)} \quad (27)$$

3.6. Multi-Objective Ant-Q

MOAQ [23] was originally proposed to design water distribution irrigation networks. It is an adaptation of the Ant-Q algorithm [24] in order to deal with multiple objectives by creating f families of ants. All families share a single pheromone matrix and each of them optimizes one objective.

After one family of ants finishes building a solution, the next family considers the solutions found to constrain its search space. Thus, solutions are found according to the relative order of importance of the objectives to optimize. However, this characteristic is not present in the TSALBP-1/3 because there is not an order of importance regarding the number of stations and the maximum area, that is, both objectives are equally important.

In Ant-Q, as well as in MOAQ, the pheromone matrix τ contains Ant-Q values that favour the probability of finding good solutions. Hence, the transition rule that defines the ant's behaviour is determined as follows,

$$j = \begin{cases} \underset{j\in\Omega}{\arg\max}\{[\tau_{ij}]^{\alpha}[\eta^f_{ij}]^{\beta}\}, & \text{if } q > q_0 \\ z, & \text{otherwise} \end{cases} \quad (28)$$

where $q \sim U(0, 1)$ is a random number sampled from a uniform distribution, q_0 decreases with each iteration according to $q_0(t) = q_0(t-1)\cdot h/q_{max}$ (initially $q_0 = q_{max}$), and q_{max} and $h \in [0, 1]$ are predefined constants. However, computing q_0 with this approach requires h and q_{max} to be set empirically considering that higher values of q_{max} lead to lower values of q_0 in fewer iterations, and thus resulting in an increase of intensification. Now, assuming $h = 0.9$ and $q_{max} = 1$, in 50 iterations $q_0 = 0.00515$ which means that ants will be likely to perform a local search with probability 99.84%. This could potentially lead to a waste of computational resources when more than 50 iterations are performed because all the ants in the family will converge to the solution that maximizes the Ant-Q values and the respective heuristic function. Therefore, we utilize instead the simplified transition rule proposed in [24], where they show that better results can be obtained. This rule is determined as follows,

$$j = \begin{cases} \underset{j\in\Omega}{\arg\max}\{[\tau_{ij}]^{\alpha}[\eta^f_{ij}]^{\beta}\}, & \text{if } q \le q_0 \\ z, & \text{otherwise} \end{cases} \quad (29)$$

where q_0 is a predefined constant, and z is a node selected according to the following probability distribution in (30).

$$p(j) = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{u\in\Omega}[\tau_{iu}]^{\alpha}[\eta_{iu}]^{\beta}}, & \text{if } j \in \Omega \\ 0, & \text{otherwise} \end{cases} \quad (30)$$

After each ant's transition, the Ant-Q values are learnt as follows,

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho(\Delta\tau_{ij} + \gamma\max_{u\in\Omega}\{\tau_{ju}\}) \quad (31)$$

where ρ is the discount factor (analogous to evaporation), γ is the learning step size, and $\Delta\tau_{ij}$ starts at zero and then is computed when the ant has built a complete solution.

Once all ants have built their complete solutions, only those with a non-dominated solution S in the current iteration will be rewarded according to (31) using $\Delta\tau_{ij}$ as follows,

$$\Delta\tau_{ij} = \frac{O}{\sum_{o=1}^{O} f^o(S)} \quad (32)$$

where O is the number of objectives.

3.7. Multi-Objective Network Ant Colony Optimization

MONACO [25] was proposed to deal with a multi-objective network optimization problem where the policy of the network changes according to the steps of the algorithm, thus dealing with a dynamic optimization problem. In order to adapt this instance to the TSALBP-m/A, some modifications must be made. For example, ants must wait until the cycle ends before updating the pheromone trails [8].

MONACO uses as many pheromone matrices τ^o and evaporation factors ρ_o as the number of objectives to optimize, but just a single heuristic function η. The transition rule is given by the following probability distribution,

$$p(j) = \begin{cases} \dfrac{[\eta_{ij}]^{\beta}\cdot\prod_{o=1}^{O}[\tau^o_{ij}]^{\alpha_o}}{\sum_{u\in\Omega}[\eta_{iu}]^{\beta}\cdot\prod_{o=1}^{O}[\tau^o_{iu}]^{\alpha_o}}, & \text{if } j \in \Omega \\ 0, & \text{otherwise} \end{cases} \quad (33)$$

where O is the number of objectives, and α_o is the importance of pheromone matrix τ^o.

After each iteration, the pheromone trails associated to the nodes visited at least once by any ant will evaporate according to $(1-\rho_o)$. Then, all ants update the pheromone trails of their respective solutions as follows,

$$\Delta\tau^o_{ij} = \frac{Q}{f^o(S)} \quad (34)$$

where f^o is the objective function o of solution S, and Q is a constant related to the amount of pheromone to be deposited by ants. Since Q was not defined in [25], we assume $Q = 1$.

3.8. Pareto Ant Colony Optimization

PACO [26] was proposed to deal with a multi-objective portfolio selection problem. It uses one heuristic function and as many pheromone matrices as objectives. Additionally, for each pheromone matrix τ_o there is an associated weight $p_o \sim U(0, 1)$ randomly chosen.

The transition rule is determined as follows,

$$j = \begin{cases} \underset{j\in\Omega}{\operatorname{argmax}} \left\{ \left[\sum_{o=1}^{O} p_o \tau_{ij}^o \right]^{\alpha} [\eta_{ij}]^{\beta} \right\}, & \text{if } q \le q_0 \\ z, & \text{otherwise} \end{cases} \tag{35}$$

where z is a node selected according to the probability distribution in (36).

$$p(j) = \begin{cases} \dfrac{[\sum_{o=1}^{O} p_o \tau_{ij}^o]^{\alpha} [\eta_{ij}]^{\beta}}{\sum_{u\in\Omega} [\sum_{o=1}^{O} p_o \tau_{iu}^o]^{\alpha} [\eta_{iu}]^{\beta}}, & \text{if } j \in \Omega \\ 0, & \text{otherwise} \end{cases} \tag{36}$$

After each ant performs a move, its pheromone trail is updated as follows,

$$\tau_{ij}^o = (1 - \rho)\tau_{ij}^o + \rho\tau_0 \tag{37}$$

where $\tau_0 = 1$ because it was shown to find better solutions with this value rather than using smaller ones [26].

Once all ants have built their respective solution S, all the pheromone matrices are evaporated ρ and the ants with the best first and the best second solution S for each objective o are selected. Only these ants will update the pheromone trails of their respective solutions according to (38) as suggested in [8].

$$\tau_{ij}^o = \tau_{ij}^o + \rho \cdot \frac{1}{f^o(S)} \tag{38}$$

4. Experimental design

This section is structured as follows: Section 4.1 summarizes the IDEA for solving the TSALBP, Section 4.2 describes the instances considered for the experiments, and Section 4.3 reports the experimental setup and the parameters used for each algorithm.

4.1. Infeasibility Driven Evolutionary Algorithm

Besides the MOACO algorithms, we also adapt the Infeasibility Driven Evolutionary Algorithm (IDEA) proposed in [12] to compare its performance with the MOACO algorithms. IDEA was designed specifically to deal with constrained optimization problems by searching for optimum solutions near the constraint boundaries, and maintaining and evolving a small proportion of infeasible solutions. In order to adapt this algorithm to the TSALBP-m/A, we had to add a third objective which is calculated based on the relative amount of constraint cycle time violation amongst the population members.

The novelty of this algorithm is that solutions in the parent and offspring population are divided into feasible and infeasible sets. These sets are ranked separately using non-dominated sorting and crowding distance, and the solutions for the next generation are selected from these sets to maintain infeasible solutions. Besides, a predefined parameter α_I is used to maintain the set of infeasible solutions as a fraction of the size of the population.

The representation scheme of solutions in TSALBP explicitly considers ordered task-station assignments. The assignment of tasks between workstations is made according to separators within the genotype, and these will depend on the number of opened workstations for the current solution. As such, the algorithm works with a variable-length coding scheme.

The crossover operator of IDEA is based on the order-based partially mapped crossover operator which generates two offspring from any two given parents as follows [27]. First, two random cut points are selected and the genes of the first offspring are copied directly from the task genes of the first parent outside the random points. Then, the task genes inside the two cut points are copied according to the order they appeared within the second parent. Additionally, the mutation operator utilized in the algorithm consists of dividing workstations at random by inserting a separator gene. If the resulting offsprings obtained from the operators are infeasible solutions with respect to the cycle time, these are added to the population of infeasible solutions.

Table 2
Characteristics of the TSALBP instances.

Instance	c	Tasks	OS	TV (AV)
arc111-1	5755	111	40.38	568.90
arc111-2	7520	111	40.38	568.90
barthol2	85	148	25.80	83
barthold	805	148	25.80	127.60
lutz2	16	89	77.55	10
lutz3	75	89	77.55	74
mukherje	351	94	44.80	21.38
nissan	180	140	90.16	115 (3)
scholl	1394	297	58.16	277.20
weemag	56	75	22.67	13.50

4.2. Problem instances

Ten well-known problem instances available at http://www.prothius.com/TSALBP/ are used to compare the performance of the designed algorithms. These instances are: arc111-1, arc111-2, barthol2, barthold, lutz2, lutz3, mukherje, nissan, scholl, weemag. All of these instances but nissan belong to the SALBP-1 formulation, and were adapted to the TSALBP-m/A [3]. The nissan instance corresponds to a real-world problem from the Nissan factory in Barcelona (Spain).

The characteristics of the problem instances chosen present different challenges to optimization algorithms. These characteristics are described in Table 2 in terms of the order strength of the precedence graph (OS) in which a higher value indicates a higher number of precedence restrictions, the time variability (TV) computed as the ratio between the highest and lowest task operation time, and the area variability (AV) computed just like TV using the required area instead.

4.3. Experimental setup

The MOACO algorithms considered in this article utilize the same parameters regarding the importance α and β of the pheromone matrix and heuristic function, respectively. Additionally, all algorithms utilize the same evaporation rate ρ, the q_0 value for the pseudo-random-proportional rule (when applies), and the initial value of the pheromone matrices. The complete list of parameters for the MOACO algorithms and the IDEA is presented in Table 3.

We consider two different variants for each MOACO algorithm. One in which ants utilize the heuristic information and the pheromone trails to compute the transition rule (heuristic variant), and another in which ants discard the heuristic information (non-heuristic variant). We are interested in experimenting with both variants to further corroborate the findings in [4], where ants of the MACS algorithm were able to find better solutions when discarding the heuristic information.

Table 3
Parameter values of the MOACO algorithms and IDEA.

Parameter	Value
MOACO general parameters	
Number of ants	$\Gamma = 10$
Thresholds (two ants each)	$\theta = \{0.2, 0.4, 0.6, 0.7, 0.9\}$
Importance of pheromone	$\alpha = 1.0$
Importance of heuristic	$\beta = 1.0$
Evaporation	$\rho = 0.2$
Pseudo-random-proportional	$q_0 = 0.2$
Initial pheromone (except MACS)	$\tau_0 = 0.1$
Bi-Criterion Optimization (Multiple Colonies)	
Number of colonies	10
Number of ants per colony	$\Gamma = 10$
COMPETants	
Number of ants per colony	$\Gamma = 10$
MOAQ	
Learning step size	$\gamma = 0.9$
IDEA	
Population size	100
Crossover probability	$p_c = 0.8$
Mutation probability	$p_m = 0.1$
Infeasibility ratio	$\alpha_I = 0.2$

The experimentation also consists of two modalities. One in which all the algorithms are given the same amount of time to operate, and another in which they all perform the same number of iterations. We refer to these experiments as Experiments $\mathcal{A}$ and $\mathcal{B}$, respectively. The goal of Experiment $\mathcal{A}$ is to find which MOACO variant performs better when computational time is limited, thus comparing all the algorithms (including IDEA) within a convenient time window. In this case, the algorithms perform 10 repetitions of 15 min on each problem instance. Differently, the goal of Experiment $\mathcal{B}$ is to find out which variant is better when both perform the same number of iterations regardless of the computational time required. In this case, the algorithms perform the same number of repetitions with the average number of iterations performed by the heuristic variant in Experiment $\mathcal{A}$ to obtain the results within a reasonable amount of time.

The results from both experiments allow us to compare the algorithms in terms of the quality of the solutions found, and the efficiency and computational time required to do so. As such, the *quality* is determined by the multi-objective performance indicators, the *efficiency* by comparing the average number of iterations performed (Experiment $\mathcal{A}$), and the *computational time* by comparing the average number of seconds required (Experiment $\mathcal{B}$). Additionally, results from both experiments provide better insights on the quality of the heuristic functions.

The true Pareto front is not known for any of the problem instances approached. Therefore, we estimate for each problem a pseudo-optimal Pareto front by merging all the approximations obtained with the algorithms at every repetition in both Experiments $\mathcal{A}$ and $\mathcal{B}$, and remove the dominated solutions therein. As such, the pseudo-optimal Pareto front becomes our best approximation to the optimal one. Henceforth, the hyper-volume and epsilon indicators are computed for each algorithm as the average of their respective indicators at every repetition with respect to the pseudo-optimal Pareto front. Likewise, the coverage indicator is computed utilizing the pseudo-optimal Pareto front and the threshold described in Section 2.4.3.

5. Results and discussions

The results and discussions are presented as follows. Firstly, we rank the algorithms according to the results from the performance indicators. Secondly, we address their differences with respect to time and iterations according to Experiments $\mathcal{A}$ and $\mathcal{B}$, respectively. Finally, we perform statistical tests between the performance indicators of the top-4 ranked MOACO algorithms.

Hereafter, we abbreviate and refer to the heuristic variant of the different MOACO algorithms as follows: (`bic-0`) Bi-Criterion Optimization with a Single Colony, (`bic-1`) Bi-Criterion Optimization with Multiple Colonies and Update by Origin, (`bic-2`) Bi-Criterion Optimization with Multiple Colonies and Update by Origin, (`comp`) COMPETants, (`macs`) MACS, (`moaq`) MOAQ, (`mona`) MONACO, and (`paco`) PACO. Regarding the non-heuristic variants, we refer to these by adding a star (*) to the heuristic abbreviations.

5.1. Ranking of the algorithms

The large number of results obtained from the experimental design led us to present an overall summary rather than the detailed set of results for the sake of comprehensibility. Even so, the detailed results are available upon request. The overall picture obtained from the experimentation is presented in Table 4, where all the algorithms are ranked according to the different performance indicators in each of the experiments performed (values on the left-hand side of the table). Their average rank is also computed to provide a more robust ranking between the algorithms (values on the right-hand side).

The best-performing algorithms are those whose ranking is within the top-4 according to either of the performance indicators or experiment modalities. These algorithms are the heuristic and non-heuristic variants of `bic-1` and `bic-2`, `moaq`*, and `comp`*. The absolute worst-performing algorithm under Experiment $\mathcal{A}$ was IDEA, and given such results we decided not to include it in Experiment $\mathcal{B}$. The average values obtained from its performance indicators were $\bar{I}_H = 0.65$, $\bar{I}_\epsilon = 1.36$, and $\bar{I}_C^\theta = 0.01$, when the same indicators for the worst performing MOACO algorithm were $\bar{I}_H = 0.90$ (`macs`), $\bar{I}_\epsilon = 1.24$ (`macs`), and $\bar{I}_C^\theta = 0.17$ (`paco`).

Furthermore, the ranking of the algorithms in Experiment $\mathcal{A}$ shows that the non-heuristic variants always rank higher than their heuristic counterparts regardless of the performance indicators. These findings confirm those in [4], where the experimental design had the algorithms perform as many iterations as possible within a time-limited setup. However, the ranking of the algorithms in Experiment $\mathcal{B}$ shows cases in I_C where the heuristic variants rank higher than their respective non-heuristic counterparts (such as `bic-1` and `paco` in I_H, `bic-2` in I_ϵ, and `mona` and `paco`). Besides, notice that the ranking differences between the heuristic and non-heuristic variants in Experiment $\mathcal{B}$ are much more reduced than those in Experiment $\mathcal{A}$. Consequently, in the next section, we further explore the number of iterations and computational time required by the different variants according to Experiments $\mathcal{A}$ and $\mathcal{B}$, respectively.

5.2. Analysis of the MOACO algorithms according to time and iterations

The algorithms in Experiment $\mathcal{A}$ iterate for 15 min in each repetition, whereas those in Experiment $\mathcal{B}$ perform a certain number of iterations regardless of the computational time they require to finish. The number of iterations each algorithm performs in Experiment $\mathcal{B}$ corresponds to the average number performed by the heuristic variants in Experiment $\mathcal{A}$. Thus, Experiment $\mathcal{A}$ evaluates the performance of the algorithms according to their computational efficiency, whereas Experiment $\mathcal{B}$ determines whether their computational complexity is worthwhile.

In Experiment $\mathcal{A}$, the non-heuristic variants significantly outperformed the heuristic ones in every problem instance. However, a decisive factor to such a performance is the expensive computational cost involved in computing the heuristic functions. Within 15 min, the non-heuristic variants were able to perform about twice

Table 4
Ranking of the algorithms according to their performance indicators.

r	I_H		I_ϵ		I_C		$\mathcal{A}$		$\mathcal{B}$		$\mathcal{A}, \mathcal{B}$	
	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{A}$	$\mathcal{B}$	moaco	$\bar{r}$	moaco	$\bar{r}$	moaco	$\bar{r}$
Top												
1	bic-1*	bic-1	moaq*	moaq*	bic-1*	bic-1	bic-1*	1.7	bic-1	1.7	bic-1*	1.8
2	bic-2*	bic-1*	macs*	bic-1*	bic-2*	bic-1*	bic-2*	2.7	bic-1*	2.0	bic-2*	3.2
3	comp*	bic-2*	bic-1*	bic-1	comp*	bic-2*	moaq*	3.7	bic-2*	3.7	bic-1	3.8
4	bic-1	bic-2	bic-2*	bic-2	macs*	bic-2	comp*	4.0	bic-2	4.0	moaq*	4.3
Middle												
5	moaq*	comp*	mona*	bic-2*	moaq*	comp*	macs*	4.3	moaq*	5.0	comp*	4.8
6	bic-2	comp	comp*	mona*	bic-1	comp	bic-1	6.0	comp*	5.7	bic-2	5.3
7	macs*	moaq*	bic-2	comp*	bic-2	moaq*	bic-2	6.7	comp	7.0	macs*	6.3
8	mona*	bic-0*	bic-1	macs*	comp	macs*	mona*	7.3	macs*	8.3	mona*	8.0
9	bic-0*	macs*	bic-0*	comp	mona*	mona	bic-0*	9.7	mona*	8.7	comp	8.3
10	comp	mona*	mona	mona	mona	mona*	comp	9.7	bic-0*	10.0	bic-0*	9.8
11	paco*	bic-0	comp	bic-0*	bic-0*	bic-0*	mona	10.7	mona	10.3	mona	10.5
12	mona	mona	paco*	paco*	macs	macs	paco*	12.3	bic-0	13.0	bic-0	13.3
Bottom												
13	bic-0	paco	paco	paco	bic-0	bic-0	bic-0	13.7	paco	13.7	paco*	13.3
14	paco	moaq	moaq	moaq	paco*	moaq	paco	14.3	moaq	14.0	paco	14.0
15	moaq	paco*	bic-0	bic-0	moaq	paco	macs	14.7	paco*	14.3	moaq	14.3
16	macs	macs	macs	macs	paco	paco*	moaq	14.7	macs	14.7	macs	14.7
17	idea		idea		idea		idea	17.0			idea	17.0

$\bar{r}$: Average ranking.

the number of iterations that the heuristic variants did. This can be observed in Table 5, which shows the number of iterations performed by the heuristic variants next to the inverse ratio indicating the factors by which these were outnumbered using the non-heuristic ones. While such a difference in iterations depends on the implementation, computing the heuristic functions does indeed involve an additional computational cost which will reduce the number of iterations to be performed.

Furthermore, Table 5 shows that the most computationally expensive algorithms are bic-1 and bic-2, followed at a distance by comp, all of which manage to perform the least number of iterations. Such a difference is expected due to the larger number of ants utilized by bic-1 and bic-2 ($\Gamma = 100$) and comp ($\Gamma = 20$) with respect to the remaining algorithms ($\Gamma = 10$). Nonetheless, the quality of the results provided by the non-heuristic variants of these algorithms ranked them within the top-5, and the heuristic ones ranked at least on the upper-middle part except for comp. Thus, their additional computational cost does actually improve the quality of results. Differently, the most computationally efficient algorithms are mona, macs and paco, from which the non-heuristic variants rank within the upper-middle part (except for paco in bottom-4), but their heuristic ones rank in the bottom-4 (except for mona in lower-middle).

In Experiment $\mathcal{B}$, Table 6 shows that the non-heuristic variants generally outperformed the heuristic ones as well, but the differences were much more reduced up to the point of having the heuristic variants of bic-1 and bic-2 within the top-4, and the comp one right after its non-heuristic variant. Even more, the bic-1 ranked higher than its non-heuristic one. Hence, it can be seen that incorporating the heuristic functions does improve the performance of the algorithms to a certain extent. However, the heuristic variants are often not better than their counterparts and, more importantly, their computational cost is twice as much. Therefore, the additional guidance provided by the heuristic functions in TSALBP-m/A is generally not worthwhile in terms of quality of results and much less in computational complexity. Nonetheless, statistical tests between the results will help consolidating these findings.

5.3. Statistical significance tests between the top-4 algorithms

Statistical significance tests are performed on the performance indicators of the average top-4 algorithms in both (heuristic and non-heuristic) variants under Experiments $\mathcal{A}$ and $\mathcal{B}$. The goal is to determine whether the differences between the best-performing algorithms are statistically significant regardless of the optimization problem. To this end, we utilize the pairwise Wilcoxon signed-rank test at a significance level $\alpha = 0.05$ between every pair of algorithms matching their performance indicators at each repetition on all problem instances. We prefer the Wilcoxon test because it does not assume the normality of the samples and has shown to be helpful to analysing the performance of metaheuristics [28].

The results of these tests on the hyper-volume indicators for Experiments $\mathcal{A}$ and $\mathcal{B}$ are presented in Tables 7 and 8, respectively, those **on** the epsilon indicators in Tables 9 and 10,

Table 6
Average time required by the non-heuristic variants and corresponding ratios to the heuristic ones (Experiment $\mathcal{B}$).

moaco	arc111-1		arc111-2		barthol2		barthold		lutz2		lutz3		mukherje		nissan		scholl		weemag	
	s^*	r	s^*	r	s^*	r	s^*	r	s^*	r	s^*	r	s^*	r	s^*	r	s^*	r	s^*	r
bic-0*	4.6	1.9	4.5	1.9	4.2	2.1	4.1	2.2	6.5	1.4	6.2	1.4	3.1	2.8	5.2	1.7	4.3	2.1	4.8	1.8
bic-1*	4.6	1.9	4.5	2.0	3.8	2.3	3.6	2.4	6.5	1.4	6.1	1.5	2.9	3.1	4.7	1.9	3.6	2.5	5.0	1.7
bic-2*	4.5	1.9	4.6	1.9	3.7	2.4	3.6	2.5	6.7	1.3	6.4	1.4	2.7	3.3	4.7	1.9	3.5	2.5	4.8	1.8
comp*	4.8	1.9	4.7	1.9	3.9	2.3	4.1	2.2	6.9	1.3	6.5	1.4	3.0	3.0	6.2	1.4	3.6	2.4	4.8	1.8
macs*	4.7	1.9	4.5	1.9	3.4	2.6	3.2	2.7	6.6	1.4	5.9	1.5	2.5	3.6	5.7	1.5	3.2	2.7	4.0	2.1
moaq*	4.5	2.0	4.4	2.0	3.6	2.5	3.5	2.5	6.9	1.3	6.5	1.4	2.9	3.0	5.6	1.6	3.1	2.8	4.4	1.9
mona*	7.0	1.3	6.4	1.3	3.5	2.5	3.4	2.5	7.3	1.2	5.8	1.6	2.5	3.5	5.2	1.7	5.0	1.7	3.8	2.2
paco*	4.1	2.1	3.9	2.4	3.7	2.4	3.0	2.9	6.8	1.3	6.2	1.5	2.2	4.2	5.3	1.7	3.2	2.7	2.6	3.4

Number of seconds: $s^* \times 100$. Ratio: $r = s/s^*$.

Table 7
Significance tests on the hyper-volume indicator for both variants of the average top-4 MOACO algorithms (Experiment $\mathcal{A}$).

moaco	r	c	A	A*	B	B*	C	C*	D	D*	$\bar{I}_H$
bic-1	4	A	•	–	+	–	+		+		0.93
	1	A*	+	•	+		+	+	+	+	0.94
bic-2	6	B	–	–	•	–	+		+		0.92
	2	B*	+		+	•	+	+	+	+	0.94
comp	7	C	–	–	–	–	•		+		0.91
	3	C*	+		–		–		•	+	0.93
moaq	8	D	–	–	–	–	–	–	•	–	0.89
	5	D*	–	–	–	–		–	+	•	0.93

r: ranking in terms of $\bar{I}_H$. c: code for MOACOs. * non-heuristic variant.

Table 8
Significance tests on the hyper-volume indicator for both variants of the average top-4 MOACO algorithms (Experiment $\mathcal{B}$).

moaco	r	c	A	A*	B	B*	C	C*	D	D*	$\bar{I}_H$
bic-1	1	A	•				+		+	+	0.94
	2	A*		•			+		+	+	0.94
bic-2	4	B			•		+		+	+	0.94
	3	B*				•	+		+	+	0.94
comp	6	C	–	–	–	–	•		+		0.92
	5	C*						•	+	+	0.93
moaq	8	D	–	–	–	–	–	–	•	–	0.90
	7	D*	–	–	–	–		–	+	•	0.92

r: ranking in terms of $\bar{I}_H$. c: code for MOACOs. * non-heuristic variant.

Table 9
Significance tests on the epsilon indicator for both variants of the average top-4 MOACO algorithms (Experiment $\mathcal{A}$).

moaco	r	c	A	A*	B	B*	C	C*	D	D*	$\bar{I}_\epsilon$
bic-1	6	A	•	–		–			+	–	1.11
	2	A*	+	•	+		+		+		1.10
bic-2	5	B		–	•	–			+	–	1.11
	3	B*	+		+	•	+		+		1.10
comp	7	C		–		–	•		+	–	1.12
	4	C*						•	+		1.11
moaq	8	D	–	–	–	–	–	–	•	–	1.14
	1	D*	+		+		+		+	•	1.10

r: ranking in terms of $\bar{I}_\epsilon$. c: code for MOACOs. * non-heuristic variant.

Table 10
Significance tests on the epsilon indicator for both variants of the average top-4 MOACO algorithms (Experiment $\mathcal{B}$).

moaco	r	c	A	A*	B	B*	C	C*	D	D*	$\bar{I}_\epsilon$
bic-1	3	A	•						+		1.11
	2	A*		•					+		1.11
bic-2	4	B			•				+		1.12
	5	B*				•			+		1.12
comp	7	C					•		+		1.12
	6	C*						•	+		1.12
moaq	8	D	–	–	–	–	–	–	•	–	1.14
	1	D*							+	•	1.11

r: ranking in terms of $\bar{I}_\epsilon$. c: code for MOACOs. * non-heuristic variant.

Table 11
Significance tests on the thresholded coverage indicator for both variants of the average top-4 MOACO algorithms (Experiment $\mathcal{A}$).

moaco	r	c	A	A*	B	B*	C	C*	D	D*	$\bar{\bar{I}}_C^\theta$
bic-1	5	A	•	–		–			+		0.54
	1	A*	+	•	+		+		+	+	0.69
bic-2	6	B		–	•	–			+		0.52
	2	B*	+		+	•			+		0.67
comp	7	C		–			•		+		0.46
	3	C*						•	+		0.59
moaq	8	D	–	–	–	–	–	–	•	–	0.20
	4	D*		–					+	•	0.55

r: ranking in terms of $\bar{\bar{I}}_C^\theta$. c: code for MOACOs. * non-heuristic variant.

own column. A similar performance between x and y is shown as blank. For all the indicators, a significantly better performance of x with respect to y is shown as '+' in the table whereas a significantly worse performance is shown as '–'.

The statistical significance on the different performance indicators show the following two patterns. On the one hand, the top-3 algorithms under Experiment $\mathcal{A}$ are all composed of non-heuristic variants whose performance differences are generally not significant between them. However, such algorithms are surely better than the remaining ones. On the other hand, the top-3 algorithms under Experiment $\mathcal{B}$ all show non-significant differences, and in cases such as the epsilon indicator, the differences are not even significant between the top-6 algorithms. These results show that the heuristic information does indeed provide useful guidance to the algorithms, but at a very high computational cost that is just not worth it when considering that similar or even better results can be obtained with the non-heuristic algorithms at about half of their computational cost.

The design of these statistical tests may disregard special aspects of the algorithms that might have been better suited to approach problems with certain characteristics. However, the advantage is that these statistical tests indicate that the best-performing algorithms are the non-heuristic variants of `bic-1` and `bic-2`

Table 12
Significance tests on the thresholded coverage indicator for both variants of the average top-4 MOACO algorithms (Experiment $\mathcal{B}$).

moaco	r	c	A	A*	B	B*	C	C*	D	D*	$\bar{\bar{I}}_C^\theta$
bic-1	1	A	•					+	+	+	0.66
	2	A*		•					+	+	0.63
bic-2	4	B			•				+	+	0.62
	3	B*				•		+	+	+	0.62
comp	6	C					•		+		0.50
	5	C*	–			–		•	+		0.54
moaq	8	D	–	–	–	–	–	–	•	–	0.23
	7	D*	–	–	–	–			+	•	0.46

r: ranking in terms of $\bar{\bar{I}}_C^\theta$. c: code for MOACOs. * non-heuristic variant.

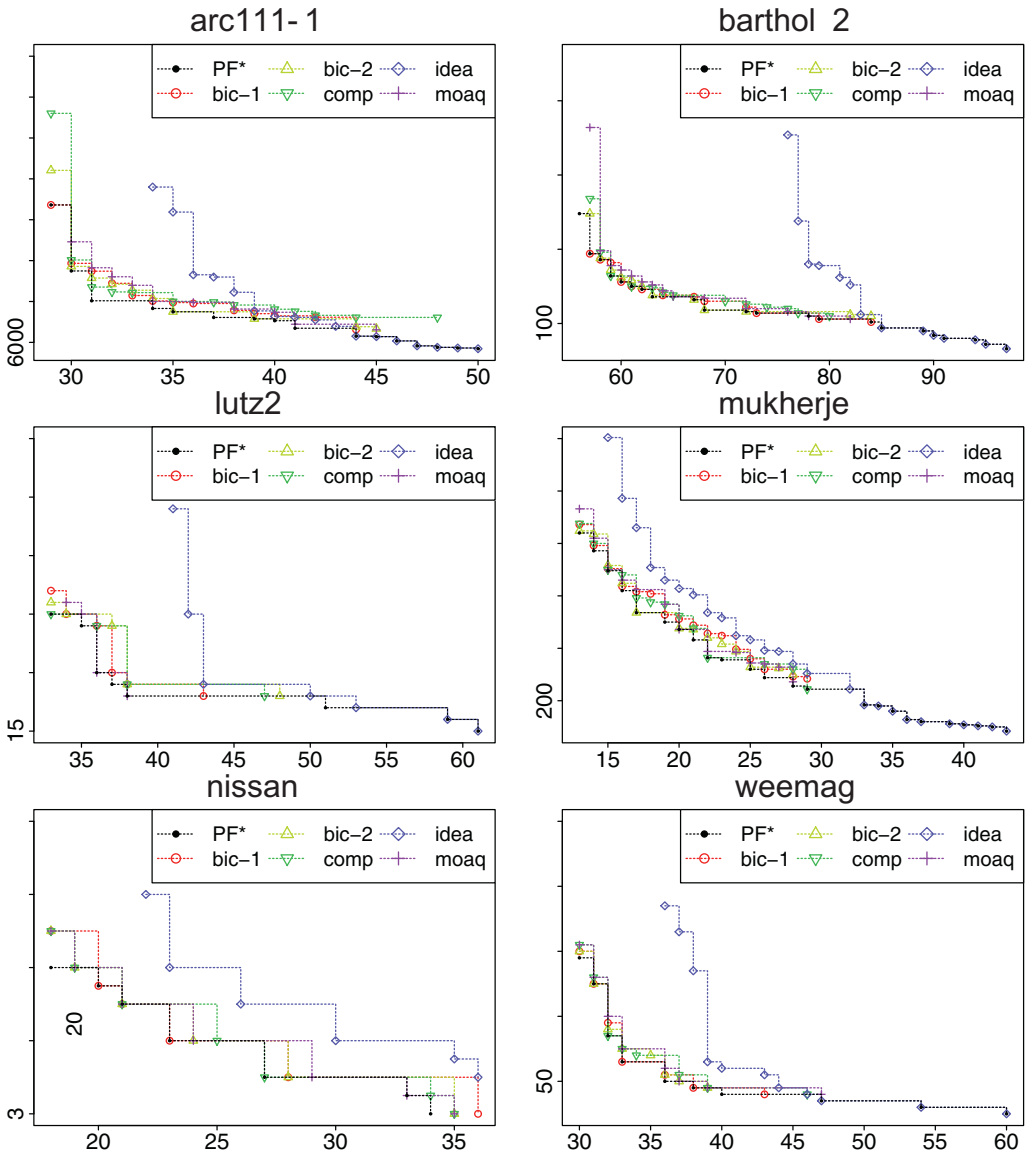

Fig. 3. Pareto fronts obtained with the non-heuristic variants of the average top-4 algorithms and IDEA in Experiment *A*. The vertical axis indicates the physical area required, while the horizontal axis indicates the number of workstations. The pseudo-optimal Pareto front is depicted by PF*.

regardless of the problem instance we approached. As stated before, these two algorithms have a much larger number of ants than the others in consideration, and it provides them with a greater diversity of solutions which ultimately leads to better results than those of the other algorithms. Furthermore, these results also hint that the other algorithms may be reaching early convergence and hence wasting a significant amount of computational processing. However, this is a conjecture that needs to be supported on a further analysis of the convergence properties for the different algorithms.

5.4. Global analysis. Advantages and disadvantages of the algorithms

Lastly, we present a final analysis and a list of advantages and disadvantages of all the considered algorithms. This global analysis results from the performance indicators of the study and it is further supported in Fig. 3, where the Pareto front approximations obtained by IDEA in some of the problem instances significantly contrast with those obtained with the average top-4 MOACO algorithms.

The main characteristics that contribute to the superior performance of the top-4 algorithms (i.e. `bic-1`, `bic-2`, `moaq`*, and `comp`*) are having a larger number of ants to increase the diversity of the colony and therefore the chances to find better solutions, and having a different pheromone matrix for each objective to prevent the loss of information incurred when mixing different pheromones into just one matrix. The best-performing algorithms were the non-heuristic variants of `bic-1` and `bic-2`. Specifically, we attribute the performance of the these two variants to the larger number of ants ($\Gamma = 100$) which are either implicitly or explicitly directed towards different regions of the Pareto fronts. Moreover, we also consider important to have different pheromone matrices for the objectives in order to prevent the loss of information about them when aggregating preferabilities into a single pheromone matrix.

Another trait that contributed to improve the performance of `moaq` was to reinforce the pheromone matrix utilizing the heuristic functions. Similarly, we attribute the performance of `comp`* to the use of a larger number of ants $\Gamma = 20$ than the other algorithms. Lastly, the aspect of `moaq`* which we consider important towards its performance is

with the heuristic functions of those ants with non-dominated solutions according to (32).

Taking into account the global performance of the algorithms according to all the phases of the experimentation we can also remark the main disadvantages of the inferior algorithms. The main disadvantage of paco is that the pheromone matrices are only updated by the best two ants for each objective. Thus, the pheromone matrices will not contain enough information and we expect ants to reach stagnation rather early due to the lack of diversity. Regarding macs and moaq, the deterioration of their performance is directly related to the usage of the heuristic information because their non-heuristic variants are either within the top-4 (moaq*) or upper-middle part of the ranking (macs*). The main disadvantage of macs is that it utilizes two heuristic matrices and just a single pheromone one, thus relying too much on the heuristic functions and presenting a certain loss of information about the objectives in the pheromone matrix.

Generally and in view of the Pareto fronts of Fig. 3, IDEA is able to find dominant solutions favouring the reduction of the required physical area at the cost of having more workstations, and such solutions make up the right-most part of the pseudo-optimal Pareto fronts of all the problem instances except nissan. Still, even with such a presence within the pseudo-optimal Pareto front, the performance indicators rank IDEA as the worst-performing algorithm. The main disadvantage we find in IDEA is its higher computational cost due to the addition of the third objective and the additional population of infeasible solutions which is maintained and evolved throughout the search. Certainly, the MOACO algorithms are not only conceptually simpler, but also more computationally efficient.

6. Conclusions and future work

The configuration of an assembly line is a hard combinatorial optimization problem to which finding the optimum solution might be infeasible or even impossible. However, finding good solutions is still of great value to managers configuring the line. To this end, we have adapted eight MOACO algorithms to solve the TSALBP-m/A by optimizing their operation with respect to the number of workstations and the maximum physical area these require, and compared their performance on ten well-known problem instances. The comparison was performed in terms of the quality of results found by two variants of the algorithms. One in which ants utilize the commonly used heuristic functions within their transition rules, and another in which such functions are just excluded. We refer to these as the heuristic and non-heuristic variants, respectively.

The non-heuristic variants always outperformed the heuristic ones in a time-limited setup where both had the same computational resources and time to perform as many iterations as possible (Experiment $\mathcal{A}$). However, such an outcome defeats the purpose of utilizing the heuristic functions because these are supposed to aid the optimization process towards finding better solutions and not otherwise. The underlying reason for such a performance was partly because the heuristic variants required more computational time to iterate, and hence they were only able to perform about half the number of iterations that the non-heuristic variants did. Notwithstanding, we prepared an additional experiment in which the variants of all the algorithms performed the same number of iterations regardless of the computational time required (Experiment $\mathcal{B}$). The quality of the results obtained with the heuristic variant was certainly improved under this experiment, but it was still worse than the quality obtained with the non-heuristic variant. Therefore, we conclude that incorporating the heuristic functions into the MOACO algorithms deteriorate their performance in TSALBP-m/A time-limited scenarios, and even neglecting such a cost, they provide no additional value to the algorithms in consideration.

The best-performing algorithms were the non-heuristic variants of Bi-Criterion Optimization with Multiple Colonies and Updates by either Origin or Region (bic-1 and bic-2), followed by the non-heuristic ones of Competing Ant Colonies (comp) and Multi-Objective Ant-Q (moaq). Despite that the heuristic variants of bic-1 and bic-2 appeared in the top-4 under Experiment $\mathcal{B}$, their results were not significantly different from those obtained with their non-heuristic counterparts. The worst-performing MOACO algorithms are the heuristic and non-heuristic variants of paco, moaq, macs, and bic-0.

Additionally, the MOACO algorithms were compared against IDEA, an evolutionary algorithm specifically design for industrial constrained optimization problems. However, the results from IDEA were significantly worse than the worst-performing MOACO algorithm. The main disadvantage of IDEA is that it has a computational cost higher than any of the MOACO algorithms. Such a cost is mainly due to having a third objective artificially created while also maintaining and evolving a population of infeasible solutions throughout the search.

Future research on this topic may consider addressing the following goals: (i) optimize the computational requirements for the heuristic functions and quantify the improvements in terms of the quality of results, (ii) experiment with different coding schemes for the TSALBP based on the bin-packing proposal of [29], and (iii) design and evaluate different heuristic functions to the TSALBP-m/A having in mind the computational cost involved.

Acknowledgements

This work was supported by the Foundation for the Advancement of Soft Computing and by the Spanish Ministerio de Economía y Competitividad under project SOCOVIFI2 (Ref. TIN2012-38525-C02-01 and TIN2012-38525-C02-02) including EDRF funding.

References

[1] A. Scholl, Balancing and Sequencing of Assembly Lines, 2nd edition, Physica-Verlag, Heidelberg, 1999.
[2] I. Baybars, A survey of exact algorithms for the simple assembly line balancing problem, Management Science 32 (8) (1986) 909–932.
[3] J. Bautista, J. Pereira, Ant algorithms for a time and space constrained assembly line balancing problem, European Journal of Operational Research (177) (2007) 2016–2032.
[4] M. Chica, Ó. Cordón, S. Damas, J. Bautista, Multi-objective constructive heuristics for the 1/3 variant of the time and space assembly line balancing problem: ACO and Random Greedy Search, Information Sciences 180 (15) (2010) 3465–3487.
[5] M. Chica, O. Cordón, S. Damas, J. Bautista, Including different kinds of preferences in a multi-objective ant algorithm for time and space assembly line balancing on different Nissan scenarios, Expert Systems with Applications 38 (2011) 709–720.
[6] M. Chica, O. Cordón, S. Damas, J. Bautista, A new diversity induction mechanism for a multi-objective ant colony algorithm to solve a real-world time and space assembly line balancing problem, Memetic Computing 3 (2011) 15–24.
[7] M. Dorigo, T. Stützle, Ant Colony Optimization, MIT Press, Cambridge, 2004.
[8] C. García-Martínez, Ó. Cordón, F. Herrera, A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP, European Journal of Operational Research 180 (1) (2007) 116–148.
[9] D. Angus, C. Woodward, Multiple objective ant colony optimisation, Swarm Intelligence 3 (2009) 69–85.
[10] P.R. McMullen, P. Tarasewich, Multi-objective assembly line balancing via a modified ant colony optimization technique, International Journal of Production Research 44 (1) (2006) 27–42.
[11] B. Yagmahan, Mixed-model assembly line balancing using a multi-objective ant colony optimization approach, Expert Systems with Applications 38 (10) (2011) 12453–12461.
[12] H.K. Singh, A. Isaacs, T. Ray, W. Smith, Infeasibility driven evolutionary algorithm (IDEA) for engineering design optimization, in: AI 2008: Advances in Artificial Intelligence, Springer, 2008, pp. 104–115.
[13] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 26 (1) (1996) 29–41.

[14] C.C. Coello, G. Lamont, D. van Veldhuizen, Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd edition, Genetic and Evolutionary Computation, Springer, 2007.
[15] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach, IEEE Transactions on Evolutionary Computation 3 (4) (1999) 257–271.
[16] A. Auger, J. Bader, D. Brockhoff, E. Zitzler, Theory of the hypervolume indicator: optimal μ-distributions and the choice of the reference point, in: Proceedings of the 10th ACM SIGEVO Workshop on Foundations of Genetic Algorithms, ACM, 2009, pp. 87–102.
[17] E. Zitzler, L. Thiele, M. Laumanns, C. Fonseca, V. da Fonseca, Performance assessment of multiobjective optimizers: an analysis and review, IEEE Transactions on Evolutionary Computation 7 (2) (2003) 117–132.
[18] L. Sánchez, J.R. Villar, Obtaining transparent models of chaotic systems with multi-objective simulated annealing algorithms, Information Sciences 178 (2008) 950–970.
[19] S. Iredi, D. Merkle, M. Middendorf, Bi-criterion optimization with multi colony ant algorithms, in: First International Conference on Evolutionary Multi-criterion Optimization, Lecture Notes in Computer Science 1993, Springer, 2001, pp. 359–372.
[20] K. Doerner, R.F. Hartl, M. Reimann, Are COMPETants more competent for problem solving? — The case of a multiple objective transportation problem, Central European Journal of Operation Research 11 (2) (2003) 115–141.
[21] B. Barán, M. Schaerer, A multiobjective ant colony system for vehicle routing problem with time windows, in: 21st IASTED International Conference, Innsbruck (Germany), 2003, pp. 97–102.
[22] M. Dorigo, L.M. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 53–66.
[23] C.E. Mariano, E. Morales, MOAQ an Ant-Q algorithm for multiple objective optimization problems, in: Genetic and Evolutionary Computation Conference, GECCO'99, Orlando, FL, USA, 1999, pp. 894–901.
[24] L.M. Gambardella, M. Dorigo, Ant-Q: a reinforcement learning approach to the traveling salesman problem, in: 12th International Conference on Machine Learning, Tahoe City, CA, USA, 1995, pp. 252–260.
[25] P. Cardoso, M. Jesús, A. Márquez, MONACO – Multi-Objective Network Optimisation Based on an ACO, in: X Encuentros de Geometría Computacional, Seville (Spain), 2003.
[26] K. Doerner, W.J. Gutjahr, R.F. Hartl, C. Strauss, C. Stummer, Pareto ant colony optimization: a metaheuristic approach to multiobjective portfolio selection, Annals of Operations Research 131 (1) (2004) 79–99.
[27] P.W. Poon, J.N. Carter, Genetic algorithm crossover operators for ordering applications, Computers & Operations Research 22 (1) (1995) 135–147.
[28] S. García, D. Molina, M. Lozano, F. Herrera, A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization, Journal of Heuristics 15 (6) (2008) 617–644.
[29] J. Levine, F. Ducatelle, Ant colony optimisation and local search for bin packing and cutting stock problems,