Information Sciences 180 (2010) 3465–3487

Contents lists available at ScienceDirect

Information Sciences

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

Multiobjective constructive heuristics for the 1/3 variant of the time and space assembly line balancing problem: ACO and random greedy search

Manuel Chica [a,*], Óscar Cordón [a], Sergio Damas [a], Joaquín Bautista [b,c]

[a] European Centre for Soft Computing, Mieres, Spain
[b] Universitat Politècnica de Catalunya, Barcelona, Spain
[c] Nissan Chair, Barcelona, Spain

ARTICLE INFO

Article history:
Received 18 December 2009
Received in revised form 10 May 2010
Accepted 28 May 2010

Keywords:
Time and space assembly line balancing problem
Ant colony optimisation
GRASP
Multiobjective optimisation
NSGA-II
Automotive industry

ABSTRACT

In this work we present two new multiobjective proposals based on ant colony optimisation and random greedy search algorithms to solve a more realistic extension of a classical industrial problem: time and space assembly line balancing. Some variants of these algorithms have been compared in order to find out the impact of different design configurations and the use of heuristic information. Good performance is shown after applying every algorithm to 10 well-known problem instances in comparison to NSGA-II. In addition, those algorithms which have provided the best results have been employed to tackle a real-world problem at the Nissan plant, located in Spain.

1. Introduction

An assembly line is made up of a number of workstations, arranged either in series or in parallel. These stations are linked together by a transport system that aims to supply materials to the main flow and to move the production items from one station to the next.

Since the manufacturing of a production item is divided into a set of tasks, one common and difficult problem is to determine how these tasks can be assigned to the stations fulfilling certain restrictions. Consequently, the aim is finding an optimal assignment of subsets of tasks to the stations of the plant. Moreover, each task requires an operation time for its execution which is determined as a function of the manufacturing technologies and the employed resources.

A family of academic problems – referred as simple assembly line balancing problems (SALBP) – was proposed to model this situation [6,46]. Taking this family as a base and adding spatial information to enrich it, Bautista and Pereira recently proposed a more realistic framework: the time and space assembly line balancing problem (TSALBP) [5]. This new framework considers an additional space constraint to become a simplified version of real-world problems. The new space constraint emerged due to the study of the specific characteristics of the Nissan automotive plant located in Barcelona, Spain

* Corresponding author.
E-mail addresses: manuel.chica@softcomputing.es (M. Chica), oscar.cordon@softcomputing.es (Ó. Cordón), sergio.damas@softcomputing.es (S. Damas), joaquin.bautista@upc.edu (J. Bautista).
URL: http://www.nissanchair.com (J. Bautista).

0020-0255/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2010.05.033

Fig. 1. An assembly line located in the industrial plant of Barcelona (Spain).

(a snapshot of an assembly line of this industry plant is shown in Fig. 1). This extended model will fit better to the latter location.

TSALBP formulations have a multi-criteria nature [10] as many real-world problems. These formulations involve minimising three conflicting objectives: the cycle time of the assembly line, the number of stations, and the area covered by these stations. However, in spite of that multiobjective nature, there is no previous proposal of a multiobjective approach to solve any of the TSALBP variants. In this paper we have selected the TSALBP-1/3 variant which tries to minimise the number and the area of stations for a given product cycle time. We have made this decision because it is quite realistic in the automotive industry where the annual production of a plant (and therefore the cycle time) is usually set by market objectives.

As in classical SALBP formulations, one of the most important aspects in TSALBP-1/3 is the set of constraints (set of precedences or cycle time limit for each station). Hence, the use of non-constructive procedures [32] is less appropriate to solve the TSALBP-1/3 than constructive metaheuristics such as ant colony optimisation (ACO) [25]. This constructive metaheuristic was inspired by the shortest path searching behaviour of various ant species. Since the initial works of Dorigo et al. [24], several researchers have developed different ACO algorithms that performed well when solving combinatorial problems such as the travelling salesman problem, the quadratic assignment problem, the resource allocation problem, telecommunication routing, production scheduling, vehicle routing, and machine learning [25,22,18,50,27,40,9]. Even the SALBP [4,8,7] and a single-objective variant of the TSALBP [5] have been solved by means of this kind of metaheuristic.

Due to the multiobjective nature of the problem and the convenience of solving it through constructive algorithms, we will work with a multiobjective ACO (MOACO) algorithm [31,2]. This family involves different variants of ACO algorithms which aim to find not only one solution, but a set of the best solutions according to several conflicting objectives. We will focus on Pareto-based MOACO algorithms which seem to be the most promising, although other MOACO algorithms exist (see [31,2]). Within the Pareto-based family, we have chosen the multiple ant colony system (MACS) [3] to solve the TSALBP-1/3 because of its good performance when solving other multiobjective combinatorial optimisation problems in comparison with the remaining Pareto-based MOACO algorithms [31].

In addition, a multiobjective random greedy search algorithm, based on the first stage of the GRASP method [28] has been designed. It follows the same constructive scheme and Pareto-based approach used in the MACS algorithm. In this way, we have been able to compare the influence of the different search behaviours of ACO and the first stage of a GRASP in the problem solving process. Different configurations and parameter settings have been considered for both algorithms. They have been compared to each other and to two baseline approaches in 10 well-known instances of the problem. These baseline approaches were based on a multiobjective random search and the state-of-the-art NSGA-II multiobjective evolutionary algorithm [20]. Furthermore, the best variants of the designed algorithms have been applied to a real-world problem instance from the Nissan industry plant in Barcelona.

This paper is structured as follows. In Section 2, the original and extended problem formulations (the SALBP and the selected variant of the TSALBP, i.e. TSALBP-1/3) and a summary of existing SALBP solution procedures are explained. In Section 3, a description of the multiobjective constructive proposals is given. The experiments used to test the performance of the algorithms, their analysis and the application to the real-world Nissan problem are described in Section 4. Finally, in Section 5, some conclusions and proposals for future work are provided.

2. Preliminaries

In this section, some preliminary information about the problem is presented. Firstly, a general view of ALB is given. The need of new realistic extensions of the simple version of the SALBP is then introduced. Finally, some existing state-of-the-art approaches to solve the SALBP are reviewed.

2.1. The assembly line balancing problem

Manufacturing of a production item is divided into a set V of n tasks. Each task j requires a positive operation time t_j for its execution. This time is determined as a function of the manufacturing technologies and the resources employed. A subset of tasks S_k ($S_k \subseteq V$) is assigned to each station k ($k = 1,2,\ldots,m$), referred to the workload of this station. Each task j can only be assigned to a single station k.

Every task j has a set of "preceding tasks" P_j which must be accomplished before starting that task. These constraints are represented by an acyclic precedence graph, whose vertices correspond to the tasks and where a directed arc $\langle i,j\rangle$ indicates that task i must be finished before starting task j on the production line (Fig. 2). Thus, task j cannot be assigned to a station that is before the one where task i was assigned.

Each station k presents a station workload time $t(S_k)$ that is equal to the sum of the tasks' lengths assigned to it. Once permanent manufacturing conditions are achieved, the items under production flow along the line at a constant rate. Then, each station k has a time c, called the cycle time, to carry out its assigned tasks. Items are then transferred to the next station in a negligible period of time, initiating a new cycle.

The cycle time c determines the production rate r of the line ($r = 1/c$) and cannot be less than the maximum station workload time: $c \geqslant \max_{k=1,2,\ldots,m} t(S_k)$.

In general, the SALBP [6,46] focuses on grouping the tasks belonging to the set V into workstations by an efficient and coherent method. In short, the goal is to achieve a grouping of tasks minimising the inefficiency of the line or its total downtime. It also has to satisfy all the constraints imposed on the tasks and stations. This classical single-model problem contains the following features:

- mass-production of a homogeneous product,
- a given production process,
- a paced line with fixed cycle time c,
- deterministic (and integral) operation times t_j,
- no assignment restrictions besides the precedence constraints,
- a serial line layout with m stations,
- every station is equally equipped with respect to machines and workers,
- a maximisation of the line efficiency.

The SALBP belongs to a general class of sequencing problems that can be seen as bin packing problems [26] with additional precedence constraints. These constraints establish an implicit order of bins, resulting in a sequence of operations, complicating the problem solving process.

2.2. The need of a space constraint: the TSALBP

The classic SALBP model is quite limited and too general for all assembly lines. In some cases, mainly in the automotive industry, we must consider space constraints before designing the plant. The need of a space constraint design can be justified as follows:

(1) The length of the workstation is limited. Workers start their work as close as possible to the initial point of the workstation, and must fulfil their tasks while following the product. They need to carry the tools and materials to be assembled in the unit. In this case, there are constraints for the maximum allowable movement of the workers. These constraints directly limit the length of the workstation and the available space.
(2) The required tools and components to be assembled should be distributed along the sides of the line. In addition, in the automotive industry, some operations can only be executed on one side of the line. It restricts the physical space where tools and materials can be placed. If several tasks requiring large areas are put together the workstation would be unfeasible.
(3) Another usual source of spatial constraints comes from the products evolution. Focusing again on the automotive industry, when a car model is replaced with a newer one, it is usual to keep the production plant unchanged. However, the new space requirements for the assembly line may create more spatial constraints.

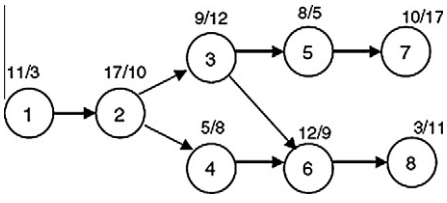

Fig. 2. A precedence graph which represents a solution for a toy-problem instance. Time and area information, separated by "/", are shown above tasks.

Based on these realistic features, a new real-like problem comes up. In order to model it, Bautista and Pereira [5] extended the SALBP into the TSALBP by means of the following formulation: the area constraint must be considered by associating a required area a_j to each task j. Every station k will require a station area $a(S_k)$, equal to the sum of the areas of all the tasks assigned to that station. The required area must not be larger than the available area A_k of the station k. For the sake of simplicity, we shall assume A_k to be identical for all the stations and denoted by A, where $A = \max_{k=1,2,\ldots,m} A_k$.

The TSALBP may be stated as: given a set of n tasks with their temporal and spatial attributes (t_j and a_j) and a precedence graph, each task must be assigned to just one station providing that:

1. all the precedence constraints are satisfied,
2. there is not any station with a workload time $t(S_k)$ greater than the cycle time c,
3. there is not any station with a required area $a(S_k)$ greater than the global available area A.

The TSALBP presents different formulations depending on which of the three considered parameters are tackled as objectives to be optimised: c, the cycle time; m, the number of stations; and A, the area of the stations. The rest of the parameters will be provided as fixed variables. The eight possible combinations result in eight different TSALBP variants. Within them, there are four multiobjective variants depending on the given fixed variable: c, m, A, or none of them. While the former three cases involve a bi-objective problem, the latter defines a tri-objective problem.

2.3. A formal description of the TSALBP constraints

As said before, restrictions play an important role in the TSALBP. In order to formally describe the TSALBP model we shall employ the following additional notation:

E_j	the earliest station to which task j may be assigned
L_j	the latest station to which task j may be assigned
UB_m	the upper bound of the number of stations. In this case, it is equal to the number of tasks
x_{jk}	a decision variable taking value 1 if task j is assigned to station k, 0 otherwise

Six different constraints can be established:

$$\sum_{k=E_j}^{L_j} x_{jk} = 1, \quad j = 1,2,\ldots,n, \tag{1}$$

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

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

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

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

$$x_{jk} \in \{0,1\}, \quad j = 1,2,\ldots,n; \ k = 1,2,\ldots,UB_m. \tag{6}$$

Constraint (1) restricts the assignment of every task to just one station, (2) limits decision variables to the total number of stations, (3) and (4) are concerned with time and area upper bounds, (5) denotes the precedence relationship among tasks, and (6) expresses the binary nature of variables x_{jk}.

2.4. The TSALBP-1/3 variant

As said, there are eight variants of the problem, four of them, multiobjective. One of these variants is the TSALBP-1/3, which consists of minimising the number of stations m and the station area A, given a fixed value of the cycle time c. We decided to work with this variant because of its realism in the automotive industry which is justified as follows:

(1) The annual production of an industry plant is usually set by some market objectives specified by the company. This fixed production rate and some other aspects such as (a) the annual working days, (b) the daily production shifts, and (c) the efficiency of the industrial processes, influence the specification of a fixed cycle time c. This means that when one of the latter conditions changes, the assembly line needs to be balanced again. These changes occur for instance if: (a) the company's chair decides to assign much more production to a factory which has lower costs than others, (b) a production reduction takes place, (c) a new shift is removed or added to the factory, (d) new staff are hired

or some part of the current staff are fired, a working days reduction arises, or (e) higher process efficiency is attained thanks to engineering projects.

(2) When we set the cycle time c, we need to search for the best number of stations m because the factory must meet the demand with the minimum number of workers. Furthermore, searching for the station area is a justified objective because it can reduce workers' movements and tool transfers.

(3) Of course, some of the theoretical values for the objective m, the number of stations, are not possible in real conditions. This is because in automotive factories the number of workers are decided in advance although changes can happen. Staff increases or decreases can also affect the production rate and its quality, being necessary a new assembly line configuration.

(4) Not only the number of stations but also some station areas, although valid in theory, may be unreachable in practice. Undesirable areas are those which are too small or too large. They can respectively generate unpleasant conditions for workers and unnecessary movements among the stations.

2.5. Heuristic procedures for the SALBP

The specific literature includes a large variety of exact and heuristic procedures as well as metaheuristics applied to the SALBP. Reviewing these approaches is out of the scope of this work. We present a brief summary and encourage the interested readers to study a seminal review in [47].

Many researchers have applied different effective solution procedures for exactly solving the SALBP (see [47]). It has resulted in about two dozen techniques mainly based on branch and bound procedures and dynamic programming approaches. Besides these techniques, several methods for reducing the effort of enumeration have been developed.

However, researchers have used constructive procedures and metaheuristics (e.g. genetic algorithms, tabu search, or simulated annealing) instead of exact methods when dealing with large SALBP instances. Some examples of these proposals are summarised as follows:

2.5.1. Constructive procedures

Most of these approaches are based on priority rules and restricted enumerative schemes [49]. Two construction schemes are relevant: (a) *station-oriented*, which starts by opening a station and selecting the most suitable task to be assigned. When the current station is loaded maximally, it is closed and the next one is opened and ready to be filled; and (b) *task-oriented*, which selects the most preferable among all available tasks and allocates it to the earliest station to which it can be assigned. Typically, priority rule-based algorithms work unidirectionally in forward direction and build a single feasible solution.

Apart from priority rules, incomplete enumeration procedures based on exact enumeration schemes are used such as Hoffmann's heuristic [35] or the truncated enumeration [46].

2.5.2. Genetic algorithms

When genetic algorithms are applied to the SALBP, there is a difficulty that has to be solved in the encoding scheme design. This difficulty is related with the feasibility of the solutions, i.e. the cycle time limit restriction and, especially, the precedence constraints.

The standard coding is based on a vector containing the labels of the stations to which the tasks $t_1,\ldots,t_n$ are assigned [1,38]. However, the existence of unfeasible solutions is a big problem in this kind of representation. Order encoding has also been used in the literature [41,44]. With this encoding, unfeasible solutions are avoided. However, we should notice that there is more than one mapping, since several sequences may lead to the same solution. Lastly, there are indirect encodings representing the solutions by coding priority values of tasks or a sequence of priority rules [33].

2.5.3. Neighbourhood search metaheuristics

In general, all local search procedures are based on shifts (a task j is moved from station k_1 to k_2) and swaps (tasks j_1 and j_2 are exchanged between different stations k_1 and k_2). The use of tabu search was proposed in [11], considering a best fit strategy (i.e., the most improving or least deteriorating move applied at each iteration), a short-term tabu list, and a frequency-based (long-term) memory. Moreover, some simulated annealing algorithms based on shifts and swaps have been proposed in the literature [34]. In [48], the SALBP-1 is tackled with one such approach when considering stochastic task times.

2.6. ACO algorithms to solve the SALBP and the TSALBP

As mentioned in the introduction, constructive algorithms and particularly ACO algorithms, are very suitable to tackle both the SALBP and the TSALBP. Bautista and Pereira [5] proposed an ACO algorithm to solve a single-objective variant of the TSALBP, TSALBP-1, which tries to minimise the number of stations m, while fixing both the cycle time c and the station area A. That proposal is based on two previous papers that are applied to the SALBP [4,8], where the authors used a priority rules procedure with an ACO and a Beam-ACO algorithm, respectively. The latter proposal was later extended in [7].

In [5], the single-objective TSALBP-1 variant was handled with an ant colony system (ACS) algorithm [23]. The heuristic information considered was built from a mixed rule of area and time information:

$$\eta_j = \frac{t_j}{c} + \frac{a_j}{A} + \frac{|F_j|}{\max_{i\in\Omega}|F_i|}, \tag{7}$$

where t_j and a_j are the time and area information for each task, normalised with their upper bounds. F_j is the set of tasks that come after task j. The third term in the formula represents a ratio between the number of successors of the task j (the cardinality of the successors set F_j) and the maximum number of successors of any eligible task belonging to the ant's feasible neighbourhood Ω.

The constructive procedure is *station-oriented*. Consequently, it is started by opening the first station and filling it with the best task selected. It will be the best task according to the pheromone trail and heuristic information (transition rule). A new station is opened when the current one is full either due to the cycle time or the area. The construction of the solution finishes when every task is assigned to a station. Although this kind of algorithm is able to generate unfeasible solutions, modifying the cycle time and the area space to create new solutions and avoiding being stuck in a local optima. We will not consider this aspect in the proposal as we will always handle feasible solutions in order to simplify the search algorithm (see Section 3).

3. Our proposal: multiobjective ACO and random greedy search algorithms for the TSALBP-1/3

In our case, a solution is an assignment of different tasks to different stations. In contrast to simpler assignment problems like bin packing [26], we have to deal with the important issue of satisfying precedence constraints. We can face the precedence problem in a proper and easy way by using a constructive approach. In the following subsections we review the basis of MACS and the selected Pareto-based MOACO algorithm. Then, we describe the two approaches based on MOACO and GRASP, starting with an overview of the common aspects and describing their specific characteristics later.

3.1. Multiple ant colony system

MACS was proposed as a variation of MACS-VRPTW [29], both based on ACS [23]. Nevertheless, MACS uses a single pheromone trail matrix τ and several heuristic information functions η^k (in this work, η^0 for the operation time t_j of each task j and η^1 for its area a_j). From now on, we restrict the description of the algorithm to the case of two objectives. In this way, an ant moves from node i to node j by applying the following transition rule:

$$j = \begin{cases} \arg\max_{j\in\Omega}\left(\tau_{ij}\cdot[\eta^0_{ij}]^{\lambda\beta}\cdot[\eta^1_{ij}]^{(1-\lambda)\beta}\right), & \text{if } q \leqslant q_0, \\ \hat{i}, & \text{otherwise}, \end{cases} \tag{8}$$

where Ω represents the current feasible neighbourhood of the ant, β weights the relative importance of the heuristic information with respect to the pheromone trail, and λ is computed from the ant index h as $\lambda = h/M$. M is the number of ants in the colony, $q_0 \in [0,1]$ is an exploitation-exploration parameter, q is a random value in $[0,1]$, and $\hat{i}$ is a node. This node is selected according to the probability distribution $p(j)$:

$$p(j) = \begin{cases} \frac{\tau_{ij}\cdot[\eta^0_{ij}]^{\lambda\beta}\cdot[\eta^1_{ij}]^{(1-\lambda)\beta}}{\sum_{u\in\Omega}\tau_{iu}\cdot[\eta^0_{iu}]^{\lambda\beta}\cdot[\eta^1_{iu}]^{(1-\lambda)\beta}}, & \text{if } j\in\Omega, \\ 0, & \text{otherwise}. \end{cases} \tag{9}$$

The algorithm performs a local pheromone update every time an ant crosses an edge $\langle i,j\rangle$. It is done as follows:

$$\tau_{ij} = (1-\rho)\cdot\tau_{ij} + \rho\cdot\tau_0. \tag{10}$$

Initially, τ_0 is calculated by taking the average costs, $\hat{f}^0$ and $\hat{f}^1$, of each of the two objective functions, f^0 and f^1, from a set of heuristic solutions by applying the following expression:

$$\tau_0 = \frac{1}{\hat{f}^0\cdot\hat{f}^1}. \tag{11}$$

However, the value of τ_0 is not fixed during the algorithm run, as usual in ACS, but it undergoes adaptation. At the end of each iteration, every complete solution built by the ants is compared with the Pareto archive P_A, which was generated till that moment. This is done in order to check if a new solution is a non-dominated one. If so, it is included in the archive and all the dominated solutions are removed. Then, τ_0' is calculated by applying the Eq. (11). The average value of each objective function is taken from the current solutions of the Pareto archive. If $\tau_0' > \tau_0$, being τ_0 the initial pheromone value, the pheromone trails are reinitialised to the new value $\tau_0 = \tau_0'$. Otherwise, a global update is performed with each solution S of the Pareto set contained in P_A by applying the following rule on its composing edges $\langle i,j\rangle$:

$$\tau_{ij} = (1-\rho)\cdot\tau_{ij} + \frac{\rho}{f^0(S)\cdot f^1(S)}. \tag{12}$$

3.2. Randomised construction procedure

Taking the greedy approaches used in [5,4] to tackle the SALBP and the TSALBP as a base, we introduce the following random elements in the construction scheme:

- A random priority rule to select a task among all the candidates. The choice is carried out at each construction step (which was already presented in [15] for the MACS algorithm).
- A novel mechanism to decide whether a station has to be closed or not.

Since the two approaches are constructive and station-oriented (see Section 2.5), the algorithms will open a station and select one task among the candidates by means of a random priority rule. Depending on each algorithm scheme, the current station will be either closed when it becomes full, as is usual in the SALBP and the TSALBP, or closed at some random point before being full in order to increase the search diversity. Then, a new station is opened to be filled. Considering this procedure and the latter two aspects we have designed two problem solving approaches. The first one is based on the MACS algorithm. The second, is inspired by the first stage of a GRASP method [28], a random greedy search algorithm. From now on, this algorithm will be called MORGA (multiobjective random greedy search algorithm).

3.3. Objective functions and Pareto-based approach

Apart from the constructive procedure, both algorithms also share some basic aspects, such as the objective definitions and the Pareto-based approach.

According to the TSALBP formulation, the 1/3 variant deals with the minimisation of the number of stations m and the area A. We can mathematically formulate the two objectives as follows:

$$f^0(x) = m = \sum_{k=1}^{UB_m} \max_{j=1,2,\ldots,n} x_{jk}, \tag{13}$$

$$f^1(x) = A = \max_{k=1,2,\ldots,UB_m} \sum_{j=1}^{n} a_j x_{jk}, \tag{14}$$

where UB_m is the upper limit for the number of stations m, a_j is the area information for task j, x_{jk} is a decision variable taking the value 1 if the task j is assigned to the station k, and n is the number of tasks.

In this work, all the multiobjective algorithms use a Pareto archive. This archive stores every non-dominated solution found during the algorithm execution. When a new solution is built it is compared with the solutions of the archive. If the new solution is non-dominated, it is included in the Pareto archive and all the solutions dominated by the new one are removed. Otherwise, the new solution is rejected.

3.4. A MACS algorithm for the TSALBP-1/3

In this section we describe the customisation on the components of the general MACS scheme to build a solution methodology.

3.4.1. Heuristic information

MACS works with two different heuristic information values, η_j^0 and η_j^1, each of them associated to one criterion. In our case, η_j^0 is related with the required operation time for each task and η_j^1 with the required area:

$$\eta_j^0 = \frac{t_j}{c} \cdot \frac{|F_j|}{\max_{i\in\Omega}|F_i|}, \tag{15}$$

$$\eta_j^1 = \frac{a_j}{UB_A} \cdot \frac{|F_j|}{\max_{i\in\Omega}|F_i|}, \tag{16}$$

where UB_A is the upper limit for the area (the sum of all tasks' areas) and the remaining variables are explained in Eq. (7). Both sources of heuristic information range the interval $[0,1]$, being 1 the most preferable.

As usual in the SALBP, tasks having a large value of time (a large duration) and area (occupying a lot of space) are preferred to be firstly allocated in the stations. Apart from the area and time information, we have added further information related to the number of successors of the task which was already used in [5]. Tasks with a larger number of successors are preferred to be allocated first.

Heuristic information is one-dimensional since it is only assigned to tasks. In addition, it can be noticed that heuristic information has static and dynamic components. Tasks' time t_j and area a_j are always fixed while the successors rate is changing through the constructive procedure. This is because it is calculated by means of the candidate list of feasible and non-assigned tasks at that moment.

We have analysed different settings for these heuristic information functions in order to find out the best possible design. As we will discuss in Section 4.3, we have studied the heuristics η_j^0 and η_j^1 with and without successors information. In addition, experiments with a MACS variant that does not take heuristic information into account have also been run.

3.4.2. Pheromone trail and τ_0 calculation

The pheromone trail information has to memorise which tasks are the most appropriate to be assigned to a station. Hence, pheromone has to be associated to a pair $(station_k, task_j)$, being $k = 1,\ldots,n$ and $j = 1,\ldots,n$. In this way, and contrary to the heuristic information, the pheromone trail matrix τ_{kj} has a bi-dimensional nature since it links tasks with stations.

In every ACO algorithm, an initial value for the pheromone trails has to be set up. This value is called τ_0 and it is normally obtained from an heuristic algorithm. We have used two station-oriented, single-objective greedy algorithms to calculate it, one per heuristic. These algorithms open the first station and select the best possible task according to their heuristic information (related either with the duration time and successors rate η_j^0, or the area and successors rate η_j^1). This process is repeated until there are no tasks that can be included because of the cycle time limit. Then, a new station must be opened. When there are no tasks to be assigned, the greedy algorithm finishes. τ_0 is then computed, using the following MACS equation, from the costs of the two solutions obtained by the greedy algorithm.

$$\tau_0 = \frac{1}{f^0(S_{\text{time}}) \cdot f^1(S_{\text{area}})}. \tag{17}$$

3.4.3. Randomised station closing scheme and transition rule

At the beginning, we decided to close the station when it was full in relation to the fixed cycle time c as usual in SALBP and TSALBP applications. We found that this scheme did not succeed because the obtained Pareto fronts did not have enough diversity (see the obtained results in [15]). Thus, we introduced a new mechanism in the construction algorithm to close the station according to a probability, given by the filling rate of the station:

$$p(\textit{closing } S_k) = \frac{\sum_{i \in S_k} t_i}{c}. \tag{18}$$

This probability distribution is updated at each construction step. A random number is uniformly generated in [0,1] after each update to decide whether the station is closed or not. If the decision is not to close the station, we choose the next task among all the candidate tasks using the MACS transition rule and the procedure goes on.

Because of the one-dimensional nature of the heuristic information, the original transition rule (see Eqs. (8) and (9)) which chooses a task among all the candidates at each step has been modified as follows:

$$j = \begin{cases} \arg\max_{j \in \Omega} \left(\tau_{kj} \cdot [\eta_j^0]^{\lambda\beta} \cdot [\eta_j^1]^{(1-\lambda)\beta} \right), & \text{if } q \leqslant q_0, \\ \hat{i}, & \text{otherwise}, \end{cases} \tag{19}$$

where $\hat{i}$ is a node selected by means of the following probability distribution:

$$p(j) = \begin{cases} \frac{\tau_{kj} \cdot [\eta_j^0]^{\lambda\beta} \cdot [\eta_j^1]^{(1-\lambda)\beta}}{\sum_{u \in \Omega} \tau_{ku} \cdot [\eta_u^0]^{\lambda\beta} \cdot [\eta_u^1]^{(1-\lambda)\beta}}, & \text{if } j \in \Omega, \\ 0, & \text{otherwise}. \end{cases} \tag{20}$$

3.4.4. Multi-colony approach

With a pure *station-oriented* procedure, intensification is too high in a Pareto front region. This region has the solutions with a small number of stations and large value of area. This is because of the constructive procedure which only closes stations when they are full. We have introduced a probability distribution according to a filling rate to solve this local convergence. It also induces more diversity in the algorithms and generate better spread Pareto fronts. Despite that, the application of this random station closing scheme carries the problem of not providing enough intensification in some Pareto front areas, since there is a low probability of filling stations completely.

Hence, there is a need to find a better intensification-diversification trade-off. This objective can be achieved by introducing different filling thresholds associated to the ants that build the solution. These thresholds make the different ants in the colony have a different search behaviour. Thus, the ACO algorithm becomes multi-colony [42,16]. In this case, thresholds are set between 0.2 and 0.9 and they are considered as a preliminary step before deciding to close a station.

Therefore, the constructive procedure is modified. We compute the station closing probability distribution as usual, based on the station current filling rate (Eq. (18)). However, only when the ant's filling threshold has been overcome, the random decision of either closing a station or not according to that probability distribution is considered. Otherwise, the station will be kept opened. Thus, the higher the ant's threshold is, the more complete the station is likely to be. This is due to the fact that there are less possibilities to close it during the construction process.

In this way, the ant population will show a highly diverse search behaviour, allowing the algorithm to properly explore the different parts of the optimal Pareto fronts by appropriately spreading the generated solutions.

3.5. MORGA

Apart from the design of the MACS algorithm, we have built a MORGA. Our diversification generation mechanism behaves similarly to a GRASP construction phase [28]. The most important element in this kind of construction is that the selection of the task at each step must be guided by a stochastic greedy function that is adapted with the pseudo-random selections made in the previous steps.

As said in Section 3.2, we introduce randomness in two processes. On the one hand, allowing each decision to be randomly taken among the best candidates, and on the other, closing the station according to a probability distribution.

We use the same constructive approach as in the MACS algorithm, with filling thresholds and closing probabilities at each constructive step. The probabilistic criterion to select the next task that will be included in the current station is changed to be only based on heuristic information. This mechanism is explained in the following paragraphs.

3.5.1. Candidate selection and heuristic-based scheme

To make a decision among all the current feasible candidate tasks we use a single heuristic value given by:

$$\eta_j = \frac{t_j}{c} \cdot \frac{a_j}{UB_A} \cdot \frac{|F_j|}{\max_{i \in \Omega} |F_i|}. \tag{21}$$

The decision is made randomly among the selected tasks in the restricted candidate list (RCL) by means of the following procedure: we calculate the heuristic value of every feasible candidate task to be assigned to the current open station. Then, we sort them according to their heuristic values and, finally, we set a quality threshold for the heuristic given by $q = \max_{\eta_j} - \gamma \cdot (\max_{\eta_j} - \min_{\eta_j})$.

All the tasks with a heuristic value η_j greater or equal than q are selected to be in the RCL. γ is the diversification-intensification trade-off control parameter. When $\gamma = 1$ there is a completely random choice inducing the maximum possible diversification. In contrast, if $\gamma = 0$ the choice is close to a pure greedy decision, with a low diversification.

3.5.2. Randomised station closing scheme

As MACS, the MORGA construction algorithm incorporates a mechanism which allows us to close a station according to a probability distribution, given by the filling rate of the station (see Eq. (18)).

As we have explained in the previous sections, this filling rate was not enough to obtain a diverse Pareto front. Consequently, we use the same MACS filling thresholds technique. The difference is that in the MACS algorithm these filling thresholds are applied in parallel following the multi-colony approach. In the case of MORGA, different thresholds are only used in isolation at each iteration.

4. Experiments

In this section we analyse the behaviour of the algorithms using unary and binary Pareto metrics, a statistical test performed over one of the binary metrics, and visual representations of the obtained Pareto fronts.[1] The used parameters and problem instances are also described in this section. Then, the experimental analysis is given.

4.1. Problem instances

We have used a total of 10 SALBP-1 instances (obtained from http://www.assembly-line-balancing.de) to run all the TSALBP-1/3 experiments. Originally, these instances only had time information but we have created their area information from the latter by reverting the task graph ($a_j = t_{n-j+1}$) to make them bi-objective (as done in [5]). In addition to these test instances, we have solved a real-world problem from a Nissan plant in Barcelona (Spain) (see http://www.nissan-chair.com/TSALBP). This real-world problem instance had specific area information for each task, so the above-mentioned method was not necessary.

These 10 well-known problem instances and the real-world one present different characteristics. They have been chosen to be as diverse as possible to test the behaviour of the algorithms and their variants when they deal with different problem conditions.[2] In Table 1 all the problem instances are shown as well as their main features values. OS refers to the order strength of the precedence graph. The higher its value, the higher the number of precedence restrictions we will find in a problem instance. TV is the time variability: the difference between the highest and the lowest task operation time. AV is

[1] From now on, we will call "true Pareto set" ("true Pareto front") the exact solution of a problem instance (which is not known here), and "Pareto set" ("Pareto front") to the set of solutions returned by an algorithm, also referred as "approximation set" in the literature.

[2] Not only the time and area information of each task influence the complexity of the problem instance, but also other factors as the cycle time limit and the order strength of the precedence graph, which actually are the most conclusive factors.

Table 1
Used problem instances.

Instance code and name		No. of tasks	OS	TV (AV)
P1	arc111 (c = 5755)	111	40.38	568.90
P2	arc111 (c = 7520)	111	40.38	568.90
P3	barthol2 (c = 85)	148	25.80	83
P4	barthold (c = 805)	148	25.80	127.60
P5	heskia (c = 342)	28	22.49	108
P6	lutz2 (c = 16)	89	77.55	10
P7	lutz3 (c = 75)	89	77.55	74
P8	mukherje (c = 351)	94	44.80	21.38
P9	scholl (c = 1394)	297	58.16	277.20
P10	wee-mag (c = 56)	75	22.67	13.50
RW	Nissan (c = 180)	140	90.16	115 (3)

OS: order strength of the precedence graph, TV and AV: time and area variability.

the same as TV but refers to the area. It is of interest in the Nissan instance because in the remainder instances the area information is obtained from the time information.

4.2. Experimental setup

In this section, a baseline multiobjective random algorithm and a state-of-the-art multiobjective evolutionary algorithm, NSGA-II [20], are given in order to set a quality threshold to test the proposals. Next, the parameter values and the considered performance metrics are presented.

4.2.1. Baseline, a basic random search algorithm

As there is no previous contribution to this problem, we are not able to compare the proposed approaches against other methods. Hence, we have designed a basic multiobjective random search algorithm based on an order encoding with separators [43].

The algorithm randomly generates a task sequence satisfying all the precedence constraints. Starting with that sequence, the algorithm needs to divide it into stations fulfilling the cycle time limit for every station it creates. To achieve that station assignment, the algorithm chooses one position to put a separator at random, but not so as to create an empty station and not to exceed the cycle time limit. The algorithm finishes when all the stations have a cycle time equal or less than the allowed one. The non-dominated solution archive and all the multiobjective mechanisms have been built as in the MACS and MORGA algorithms.

We note this is a simple algorithm. However, our aim is only to have a lower quality baseline for the approaches proposed in this paper.

4.2.2. A NSGA-II approach

As explained in Section 2.5, there are quite a lot of genetic algorithm-based proposals applied to the SALBP. However, all of them deal with a single-objective problem. The well-known NSGA-II has been considered to extend one of the existing methods to make it multiobjective since a multiobjective genetic algorithm is needed.

Hence, we have adopted the problem-dependent features of the genetic algorithm for the SALBP introduced in [44]. In short, its features can be summarised as follows:

- Coding: an order coding scheme is used. The length of the chromosome will be the number of tasks and the procedure to group tasks to form stations, is guided by fulfilling the available cycle time of each station.
- Initial population: it is randomly generated, assuring the feasibility of the precedence relations.
- Crossover: A kind of order preserving crossover is considered to ensure that feasible offspring are obtained satisfying the precedence restrictions. This family of order-based crossover operators emphasises the relative order of the genes from both parents. Two different offspring are generated from the two parents to be mated proceeding as follows. Two cutting-points are randomly selected for them. The first offspring takes the genes outside the cutting-points (i.e. from the beginning to the first cutting point and from the second cutting point to the end) in the same sequence order as in the first parent. The remaining genes, those located between the two cutting-points, are filled in by preserving the relative order they have in the second parent. The second offspring is generated in the complementary way, i.e. taking the second parent to fill in the two external parts of the offspring and the first one to build the central part. Note that, preserving the other parent genes order in the central part will guarantee the feasibility of the obtained offspring solution in terms of precedence relations. The central genes also satisfy the precedence constraints with respect to those that are in the two external parts. When resampling them in the same order they appear in the second parent, which of course encodes a feasible solution. We also manage to keep on satisfying the precedence order among them.

- Mutation: a random gene is selected, and the genes after it are randomly replaced (scrambled) assuring the precedence feasibility.

Initially, we considered the original NSGA-II design as proposed in Deb et al.'s seminal paper [20]. However, the approximations to the Pareto fronts obtained in all the developed experiments showed a significant lack of diversity and an excessive convergence to the left-most region of the objective space. Actually, we were aware of this behaviour because it is a consequence of the specific characteristics of the tackled problem. Hence, it is not the multiobjective genetic algorithm's fault (it is well known that NSGA-II has shown a large success when solving many different multiobjective numerical and combinatorial optimisation problems). As said, the presence of precedence constraints in the TSALBP-1/3 makes the use of constructive metaheuristics more appropriate to solve it than global search procedures, i.e. genetic algorithms. Moreover, the use of the order encoding makes the genotype-phenotype application not unique, thus making the search process more difficult (see Section 2.5).

Even so, we aimed to increase the diversity and spread of the obtained Pareto fronts. A study of appropriate techniques to inject diversity to the algorithm search was carried out. As a result of that study, we decided to adopt one successful and very recent NSGA-II diversity induction mechanism: Ishibuchi et al.'s similarity-based mating [37]. This method is based on setting two sets of candidates. These sets will be the couple of parents to be mated, with a pre-specified dimension α and β, respectively. The chromosomes of each set are randomly drawn from the population by a binary tournament selection. Then, the average objective vector of the first set is computed and the most distant chromosome to it, among the α candidates in the set, is chosen as the first parent. For the second parent, the most similar chromosome to the first parent is selected among the β candidates in the second set.

In [37], the authors showed how the algorithm performed better with adaptive values for the α and β parameters. They were fixed to 10 at the first stages of the evolution and then to 1 during the last stages in order to achieve a proper diversification-intensification trade-off. We ran the algorithm following the latter approach. We also considered a fixed value of 10 for both parameters, aiming to increase the algorithm's diversity as much as possible to cope with the specific characteristics of the problem. Although similar outcomes were achieved, the latter configuration induced a little more diversity in the obtained Pareto front approximations. It also showed a slightly better performance than the original NSGA-II implementation. Thus, the similarity-based mating NSGA-II algorithm will be the one considered in the experimental analysis, setting α and β parameters to 10.

Nevertheless, as shown later, the performance of this technique is unsatisfactory in properly solving the TSALBP-1/3. It does not provide the decision maker with a number of good quality assembly line design choices with a different trade-off between the number of stations and their area. We should remember that this is not due to a bad behaviour of the NSGA-II algorithm itself but to the specific problem characteristics. Thus, the representation and the non-constructive scheme are not adequate for the problem solving.

4.2.3. Parameter values

The MACS, MORGA, NSGA-II and the basic random search algorithm have been run 10 times with 10 different seeds during 900 s for each of the 11 selected problem instances. All the considered parameter values are shown in Table 2.

4.2.4. Metrics of performance

In this paper, we will consider the two usual kinds of multiobjective metrics existing in the specialised literature [51,52,19,39,17]:

- those which measure the quality of a non-dominated solution set returned by an algorithm, and
- those which compare the performance of two different multiobjective algorithms.

On the one hand, we have selected the generational distance (GD), the hypervolume ratio (HVR), and the number of different non-dominated solutions (in the objective vectors) returned by each algorithm, from the first group of metrics.

GD measures the average distance between the solutions of an approximate Pareto set P and the true Pareto set P^* by means of the following expression:

$$GD(P^*, P) = \frac{\sqrt{\sum_{p \in P} d(p)^2}}{|P|}, \tag{22}$$

where $d(p) = min\|W(p^*) - W(p)\|$ is a minimal distance between solutions of P^* and P (in the objective space).

The *HVR* can be calculated as follows:

$$HVR = \frac{HV(P)}{HV(P^*)}, \tag{23}$$

where $HV(P)$ and $HV(P^*)$ are the volume (S metric value) of the approximate Pareto set and the true Pareto set, respectively. When *HVR* equals 1, the approximate Pareto front and the true one are equal. Thus, *HVR* values lower than 1 indicate a generated Pareto front that is not as good as the true Pareto front.

Table 2
Used parameter values.

Parameter	Value
General	
Number of runs	10
Maximum run time	900 s
PC specifications	Intel Pentium™ D 2 CPUs at 2.80 GHz
Operating system	CentOS Linux 4.0 GCC 3.4.6
MACS	
Number of ants	10
β	2
ρ	0.2
q_0	0.2
Ants' thresholds	{0.2, 0.4, 0.6, 0.7, 0.9} (2 ants for each threshold)
MORGA	
γ	{0.1, 0.2, 0.3}
Diversity thresholds	{0.2, 0.4, 0.6, 0.7, 0.9}
NSGA-II	
Population size	100
Crossover probability	0.8
Mutation probability	0.1
α and β values for the similarity-based mating	10

We have to keep in mind some obstacles which make difficult the computation of these metrics because we are working with real-like problems. First, we should notice that the true Pareto fronts are not known. To overcome this problem we will consider a pseudo-optimal Pareto set, i.e. an approximation of the true Pareto set. It is obtained by merging all the (approximate) Pareto sets P_i^j generated for each problem instance by any algorithm in any run. Thanks to this pseudo-optimal Pareto set we can compute *GD* and *HVR* metrics, considering them in the analysis of results.

Besides, there is an additional problem with respect to the *HVR* metric. In minimisation problems, as ours, there is a need to define a reference point to calculate the volume of a given Pareto set. If it is not correctly fixed, the values of the *HVR* metric can be unexpected (see Fig. 3) [39]. Thus, we have defined the reference points for the as the "logical" maximum values for the two objectives. These reference points depend on each problem instance.

On the other hand, we have also considered the binary set coverage metric *C* to compare the obtained Pareto sets two by two based on the following expression:

$$C(P,Q) = \frac{|\{q \in Q;\ \exists p \in P : p \prec q\}|}{|Q|}, \tag{24}$$

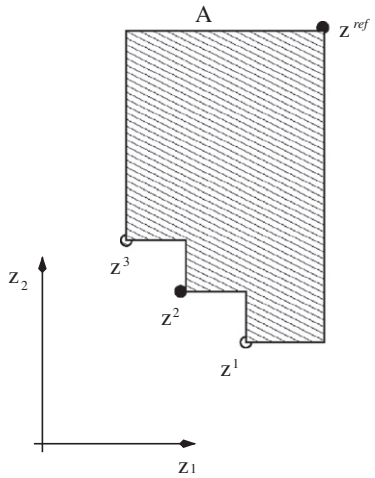

Fig. 3. Setting a non-equilibrated reference point can cause an unexpected behaviour in the *HVR* metric values [39].

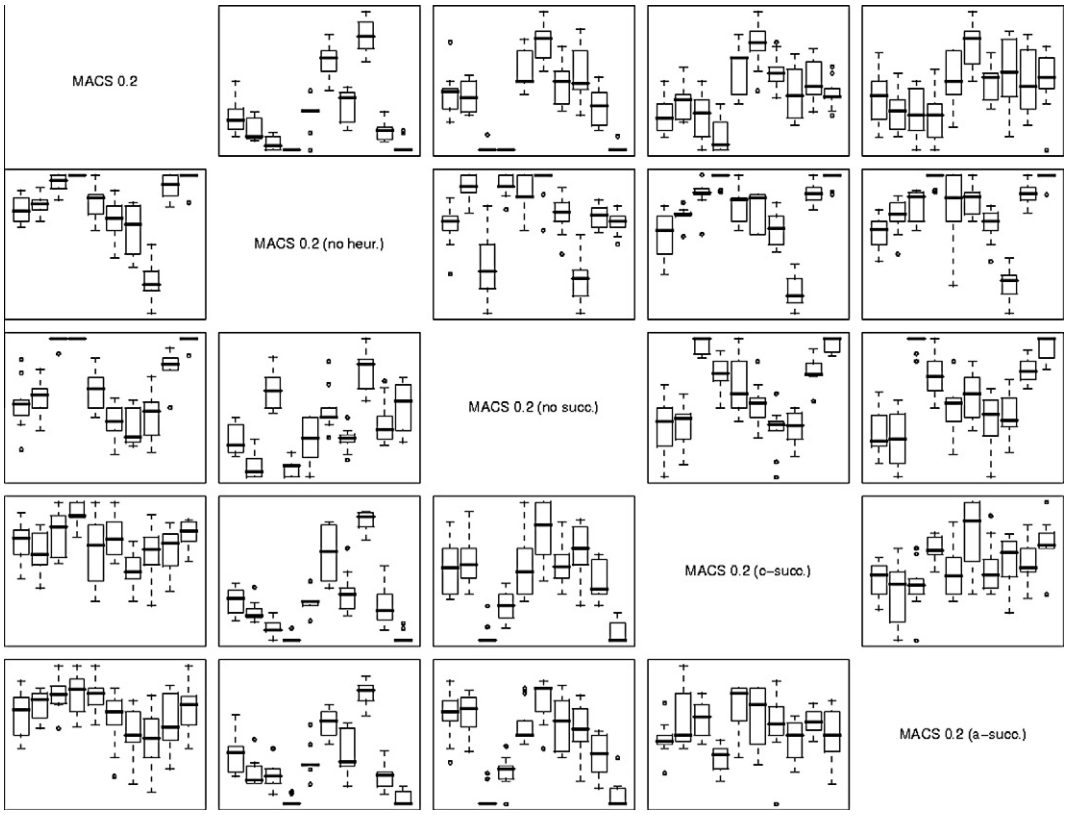

Fig. 4. *C* metric values represented by means of boxplots comparing different heuristic versions of the MACS algorithm.

where $p \prec q$ indicates that the solution p, belonging to the approximate Pareto set P, dominates the solution q of the approximate Pareto set Q in a minimisation problem.

Hence, the value $C(P,Q) = 1$ means that all the solutions in Q are dominated by or equal to solutions in P. The opposite, $C(P,Q) = 0$, represents the situation where none of the solutions in Q are covered by the set P. Note that both $C(P,Q)$ and $C(Q,P)$ have to be considered, since $C(P,Q)$ is not necessarily equal to $1 - C(Q,P)$.

We have used boxplots based on the *C* metric for showing the dominance degree of the Pareto sets of every pair of algorithms (see Figs. 4 and 7). Each rectangle contains 10 boxplots representing the distribution of the *C* values for a certain ordered pair of algorithms in the 10 problem instances (P1 to P10) and the Nissan instance. Each box refers to algorithm *A* in the corresponding row and algorithm *B* in the corresponding column, and gives the fraction of *B* covered by *A* ($C(A,B)$). The 10 considered values to obtain each boxplot correspond to the computation of the C metric on the two Pareto sets generated by algorithms A and B in each of the 10 runs. In each box, the minimum and maximum values are the lowest and highest lines, the upper and lower ends of the box are the upper and lower quartiles, and a thick line within the box shows the median.

Let us call $\overline{P_i^j}$ the non-dominated solution set returned by algorithm i in the jth run for a specific problem instance; $P_i = \overline{P_i^1} \bigcup \overline{P_i^2} \bigcup \cdots \bigcup \overline{P_i^{10}}$, the union of the solution sets returned by the 10 runs of algorithm i; and finally $\overline{P_i}$ the set of all non-dominated solutions in the P_i set.[3] Hence, the corresponding Pareto fronts will be represented graphically in different figures in order to allow an easy visual comparison of the performance of the algorithms. These graphics offer a visual information, not measurable, but sometimes more useful than numeric values. That situation becomes very clear in complex problems as that one, in which some traditional metrics seem to be deceptive.

Finally, the Mann–Whitney *U* test, also known as Wilcoxon ranksum test, will be used for a deeper statistical study of the performance of the different algorithms by considering the coverage metric. Unlike the commonly used *t*-test, the Wilcoxon test does not assume normality of the samples and it has already demonstrated to be helpful analysing the behaviour of evolutionary algorithms [30]. However, there is not a reference methodology to apply a statistical test to a binary indicator in multiobjective optimisation. Thus, we have decided to follow the procedure proposed in [45] given by: let *A* and *B* be the two algorithms to be compared. After running both algorithms just once, let $p_A(B)$ be 1 if the Pareto set generated by *A* dominates that one got by *B*, and 0 otherwise. It is considered that the Pareto set *A* dominates *B* when $C(A,B)$ is greater than a threshold value $thr \in (0.5,1)$ (in this paper we consider $thr = 0.75$). Given 10 repetitions $A_1,\ldots,A_{10}$ of *A* and $B_1,\ldots,B_{10}$ of *B*, let

[3] Note that, the pseudo-optimal Pareto set is the fusion of the $\overline{P_i}$ sets generated by every algorithm.

$P_A(B) = \frac{1}{10}\sum_{i=1}^{10} p_{A_i}(B_i)$. Note that, $P_A(B)$ corresponds to the probability that the outcomes of algorithm A dominate those of algorithm B. Hence, it becomes an indicator of the performance of A with respect to B. Following an analogous approach, let $p_B(A)$ be 1 if B dominates A (i.e. when $C(B,A) > 0.75$), and 0 otherwise. Given 10 repetitions $A_1,\ldots,A_{10}$ of A and $B_1,\ldots,B_{10}$ of B, let $P_B(A) = \frac{1}{10}\sum_{i=1}^{10} p_{B_i}(A_i)$. To know if there is a significant difference between the performance of the two algorithms, we can use a Wilcoxon test to discard the expectations of probability distributions $P_A(B)$ and $P_B(A)$ are the same. From the C metric values, $\widehat{P_A(B)}$ and $\widehat{P_B(A)}$ are computed for the considered algorithms as the average of the $P_A(B)$ and $P_B(A)$ values for the 10 problem instances. The significance level considered in all the tests to be presented is $p = 0.05$.

4.3. A deep study of heuristic information in MACS variants

A preliminary experimentation in [12] was performed to fix the value of the transition rule parameter q_0 of the MACS algorithm. Three different values were tested: 0.2, 0.5, and 0.8. The former was the one inducing the highest search diversification and it clearly provided the best performance. Here, we would like to analyse the influence of the different components of the heuristic information values in the MACS algorithm performance. To do so, we will consider different heuristic configurations over the best MACS setting: MACS 0.2 (i.e., MACS with $q_0 = 0.2$). Firstly, we have taken different combinations of the definitions of the heuristic information values η^0 and η^1 in three distinct variants of the algorithm as follows:

- MACS c-succ (c with successors information):

$$\eta_j^0 = \frac{t_j}{c} \cdot \frac{|F_j^*|}{\max_{i\in\Omega}|F_i^*|} \qquad \eta_j^1 = \frac{a_j}{UB_A}. \tag{25}$$

- MACS a-succ (a with successors information):

$$\eta_j^0 = \frac{t_j}{c} \quad \eta_j^1 = \frac{a_j}{UB_A} \cdot \frac{|F_j^*|}{\max_{i\in\Omega}|F_i^*|}. \tag{26}$$

- MACS no-succ (without successors information):

$$\eta_j^0 = \frac{t_j}{c} \quad \eta_j^1 = \frac{a_j}{UB_A}. \tag{27}$$

Besides, we have also tried to remove completely the heuristic information by considering only the pheromone trails to guide the search in MACS.

The boxplots in Fig. 4 show C metric values comparing MACS 0.2 and the new heuristic variants in the experimentation. The unary metric results for these algorithms are included in Table 3. In addition, Table 4 shows the results of the statistical test for the dominance probabilities of the MACS algorithms. Every cell of the table includes the averaged $\widehat{P_A(B)}$ value for the 10 problem instances together with a "+", "−", or "=" symbol, with a different meaning. Every symbol shows that the algorithm in that row is significantly better (+), worse (−) or equal (=) in behaviour (using the said indicator) than the one that appears in the column. For example, the pair (0.01,=) included in the first row of results (second column) must be interpreted as follows: the averaged dominance probability of MACS 0.2 with respect to MACS 0.2 no-heur is 0.01 ($\widehat{P_{MACS0.2}(\text{MACS 0.2 no-heur})} = 0.01$), and there is not any statistical significance ("=") on the performance of MACS 0.2 and MACS 0.2 no-heur.

To provide more intuitive and visual results, the graphs in Figs. 5 and 6 represent the aggregated Pareto front approximations for the P5 and P8 problem instances. We merged the outcomes of the 10 different runs performed by the process explained in Section 4.2.4 to show a visual estimation of an algorithm's performance at a glance. In addition, the solutions belonging to the pseudo-optimal Pareto front are showed linked by dashed lines in every case. The objective vectors in that line are only specifically represented by a symbol when they have been generated by any of the algorithms considered in the graph. Note that, we do not use symbols to represent the solutions of those algorithms that are not involved in the graph comparison. However, their solutions also help to compound the Pseudo-optimal Pareto front (dashed line).

We have developed the analysis grouped into three items according to the algorithms involved in the comparison:

MACS vs. Heuristic-based MACS variants. We would like to compare MACS 0.2 with the three MACS 0.2 variants which use some kind of heuristic information: MACS 0.2 no-succ, MACS 0.2 c-succ, and MACS 0.2 a-succ. In relation to the C metric, they attain "better results"[4] than MACS 0.2 in six, eight and six problem instances respectively, with the latter not dominating any of them (see Fig. 4). Similar conclusions can be drawn analysing the unary metrics' values in Table 3. The values of *HVR* show that MACS 0.2 no-succ performs better than MACS 0.2 in five, than MACS 0.2 c-succ in other five and than MACS 0.2 a-succ in seven of the 10 problem instances (with one, two and two draws, respectively). For GD, the latter three heuristic variants outperform MACS 0.2 in five, four and five instances, respectively. Thus, the general conclusion is that including successors information in

[4] When we refer to the best or better performance comparing the C metric values of two algorithms we mean that the Pareto set derived from one algorithm significantly dominates that one achieved by the other. Likewise, the latter algorithm does not dominate the former one to a high degree.

Table 3
Unary metrics for the 10 problem instances comparing the different MACS heuristic variants.

	# dif_sols	*HVR*	*GD*	# dif_sols	*HVR*	*GD*
	P1			*P2*		
A1	9.7 (1.55)	0.83 (0.01)	**508.77 (63.3)**	10.1 (1.22)	0.85 (0.01)	468.38 (61.8)
A2	**11.5 (1.8)**	**0.85 (0.01)**	647.69 (44.14)	**12.1 (1.37)**	**0.89 (0.01)**	485.44 (451.3)
A3	9.6 (1.2)	0.83 (0.01)	688.04 (96.01)	11 (1.79)	0.84 (0.02)	745.51 (734.94)
A4	10.2 (1.66)	0.84 (0.01)	619.67 (144.91)	10.2 (2.36)	0.87 (0.01)	**456.07 (82.3)**
A5	8.7 (1.49)	0.84 (0.01)	534.43 (87.91)	10.3 (0.9)	0.84 (0.01)	727.32 (130.33)
	P3			*P4*		
A1	9.9 (1.64)	0.76 (0.02)	11.01 (0.61)	11.1 (1.22)	0.67 (0.02)	37.80 (5.33)
A2	**12.8 (2.8)**	0.88 (0.01)	6.36 (1.12)	11 (0.89)	**0.92 (0.02)**	**14.15 (1.45)**
A3	9 (1.41)	**0.89 (0.01)**	**5.37 (1.12)**	**11.9 (0.54)**	0.83 (0.01)	20.99 (1.66)
A4	7.1 (1.51)	0.80 (0.02)	8.88 (1.35)	10.3 (1.55)	0.75 (0.02)	24.51 (2.35)
A5	8.8 (1.33)	0.79 (0.01)	8.54 (0.58)	10.9 (1.51)	0.77 (0.02)	22.40 (3.41)
	P5			*P6*		
A1	6.2 (0.75)	0.86 (0.02)	5.84 (1.19)	6.9 (1.45)	0.82 (0.02)	**1.06 (0.30)**
A2	**7.1 (0.3)**	**0.94 (0)**	**3.58 (1.06)**	6.7 (0.64)	**0.85 (0.02)**	1.15 (0.23)
A3	6.1 (0.7)	0.88 (0.01)	6.17 (1.72)	6.6 (1.11)	0.78 (0.02)	1.62 (0.15)
A4	6.2 (0.75)	0.87 (0.01)	6.38 (1.40)	**7 (0.89)**	0.79 (0.02)	1.40 (0.32)
A5	5.7 (0.46)	0.84 (0.02)	5.53 (1.11)	6.7 (0.46)	0.82 (0.03)	1.35 (0.22)
	P7			*P8*		
A1	8.9 (1.45)	0.80 (0.05)	7.30 (1.13)	12.1 (0.94)	0.90 (0.01)	**5.54 (1.16)**
A2	**10.3 (1.8)**	0.73 (0.03)	**5.30 (0.39)**	12.4 (1.36)	0.86 (0.01)	9.60 (1.44)
A3	8.1 (1.45)	0.65 (0.04)	5.96 (1.01)	12.2 (2.44)	0.89 (0.01)	5.91 (0.6)
A4	9.7 (1.1)	**0.96 (0.07)**	13.78 (3.70)	11.7 (1.27)	0.90 (0.01)	6.07 (1.37)
A5	9 (0.77)	0.78 (0.04)	7.64 (1.46)	**12.5 (1.02)**	**0.91 (0.01)**	5.58 (0.68)
	P9			*P10*		
A1	13.5 (0.92)	0.84 (0.01)	42.34 (8.65)	7.3 (1.19)	0.71 (0.03)	3.80 (0.41)
A2	**14.6 (2.01)**	**0.89 (0.01)**	40.59 (7.53)	8.2 (1.54)	**0.87 (0.01)**	**2.38 (0.4)**
A3	13.5 (2.42)	0.87 (0.01)	38.22 (9.66)	**8.5 (1.2)**	0.85 (0.01)	2.67 (0.45)
A4	12 (2.19)	0.84 (0.01)	**36.10 (5.98)**	7.4 (1.28)	0.74 (0.03)	3.73 (0.37)
A5	13 (1.73)	0.84 (0.01)	47.60 (12.28)	8.4 (1.02)	0.76 (0.02)	3.48 (0.22)

A1: MACS 0.2, A2: MACS 0.2 (no-heur), A3: MACS 0.2 (no-succ), A4: MACS 0.2 (a-succ), A5: MACS 0.2 (c-succ).

Table 4
Averaged dominance probability and statistical significance for the MACS variants.

	MACS 0.2	MACS 0.2 (no-heur)	MACS 0.2 (no-succ)	MACS 0.2 (c-succ)	MACS 0.2 (a-succ)
MACS 0.2	•	0.01 =	0.02 =	0.02 =	0.02 =
MACS 0.2 (no-heur)	0.3 =	•	0.19 =	0.2 +	0.22 +
MACS 0.2 (no-succ)	0.28 =	0.01 =	•	0.14 =	0.16 =
MACS 0.2 (c-succ)	0.08 =	0 –	0.04 =	•	0.05 =
MACS 0.2 (a-succ)	0.05 =	0 –	0 =	0.01 =	•

both heuristics, η^0 and η^1, is not a good option because it causes an excessive intensification. Note that, a new variant of MACS considering a different heuristic definition attains better results than the original MACS 0.2 in almost all the problem instances.

Heuristic-based MACS variants' comparison. According to the *C*, *GD*, and *HVR* metrics, we can draw three conclusions: (1) A MACS variant performs better in some problem instances and another one in some others. We cannot conclude there is a single "global best" heuristic-based MACS variant for every instance. This is supported by the results of the statistical test for the dominance probabilities (Table 4, showing how there is no significant difference in any of the comparisons). (2) Successors information is only useful in some problem instances. (3) It is a better decision linking successors information with the cycle time c_j rather than with the area information a_j.

MACS with and without heuristic information. MACS 0.2 no-heur attains better *C* metric results than MACS 0.2 and MACS 0.2 no-succ in eight problem instances, and better than MACS 0.2 c-succ and MACS 0.2 a-succ in nine of them. Globally, heu-

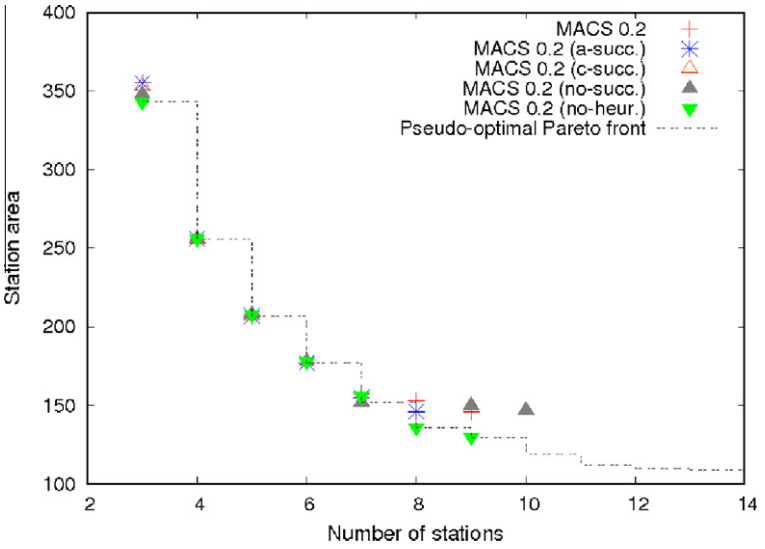

Fig. 5. Pareto fronts of the different heuristic MACS variants for the P5 instance.

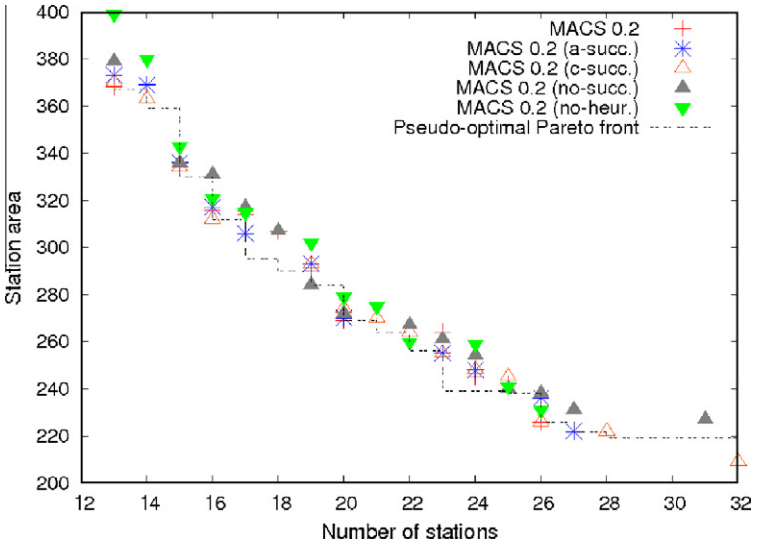

Fig. 6. Pareto fronts of the different heuristic MACS variants for the P8 instance.

ristic information is not a good guide and its use produces worse results. As an example, see the good performance of MACS 0.2 no-heur for the P5 instance in Fig. 5. This behaviour is also observed in the remaining problem instances. The only exceptions are P3 and, especially, P8 which need some kind of heuristic information to achieve convergence to the whole true Pareto front.

A further analysis can be done by means of the statistical test of Table 5. Differences in the dominance probability are significant for MACS no-heur with respect to MACS c-succ and MACS a-succ. There is not any significant difference between MACS no-heur neither with MACS nor MACS no-succ. However, we should also note the large differences existing in the averaged dominance probability values for the latter two comparisons (0.3 vs. 0.01 and 0.19 vs. 0.01, respectively), even if the Wilcoxon test does not find them to be significant.

Regarding to the unary metrics (Table 3), MACS no-heur achieves the best values in seven of the 10 problem instances for *HVR* and four for *GD*.

In summary, we can conclude that heuristic information does not help the algorithm to cover all the extension of the Pareto front and hence that MACS 0.2 no-heur is the best MACS variant. The use of heuristic information is only helpful in some problem instances, P8 and, to a lower degree, P3. In these instances, the algorithms without heuristic information are not

Table 5
Averaged dominance probability and statistical significance for the different algorithms.

	Random base	NSGA-II	MORGA 0.3	MACS 0.2 (no-heur)
Random base	•	0 =	0 –	0 –
NSGA-II	0 =	•	0.14 =	0.01 –
MORGA 0.3	0.16 +	0.19 =	•	0.12 –
MACS 0.2 (no-heur)	0.53 +	0.26 +	0.59 +	•

able to reach some areas of the true Pareto front, as can be seen in Fig. 6. Even so, in that figure we can especially notice that MACS 0.2 no-heur has a solid behaviour in comparison with the remaining algorithms.

4.4. Comparison of the best MACS variant with MORGA, multiobjective random search, and NSGA-II

After the comparison among the different MACS heuristic variants, we wanted to compare the best MACS variant (MACS 0.2 no-heur) with MORGA, the multiobjective random search, and the NSGA-II approach (the two latter algorithms were described in Section 4.2). As done for the MACS algorithm, a preliminary experimentation was performed to fix the value of the RCL parameter γ in MORGA (see Section 3.5). Three different values were tested (0.1, 0.2, and 0.3) with the latter providing the best performance (from now on, we will refer to this setting as MORGA 0.3). This shows how a larger diversification is appropriate to solve the TSALBP-1/3 with a MORGA approach.

Again, the boxplots in Fig. 7 show the *C* metric values, Table 5 comprises the results of the Wilcoxon test, and Table 6 the unary metric values resulting from the experimentation. As we did in the previous subsection, we have developed the analysis grouped in five items according to the algorithms involved in the comparison:

MORGA 0.3 vs. MACS 0.2 no-heur. If we compare MORGA with MACS no-heur, the former is clearly dominated in six problem instances according to the *C* metric values (Fig. 7). A significant difference in favour of MACS no-heur is also obtained in the statistical test (Table 5). Besides, MORGA 0.3 attains worse *GD* values in seven problem instances (Table 6). The same behaviour is observed according to the *HVR* values in the same table, MACS no-heur outperforms MORGA in six instances,

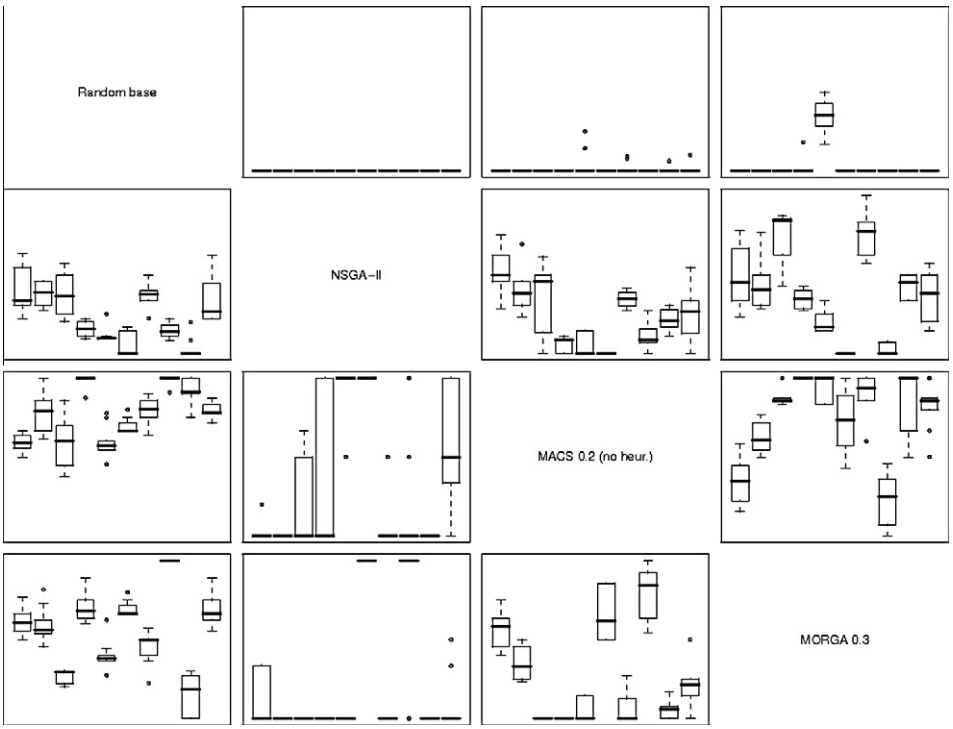

Fig. 7. *C* metric values represented by means of boxplots comparing the best MACS and MORGA variants, the random search, and the NSGA-II approach.

with an additional draw. Since MACS 0.2 no-heur is the best algorithm, we consider unnecessary to implement a new MORGA variant with different heuristic configurations.

Random search vs. MACS and MORGA best variants. In all the problem instances, MACS is "much better" than random search according to the *C* metric (Fig. 7). As MORGA's strength is lower than MACS's one, the performance improvement of this algorithm against random search is not as high as in MACS but it is clear as well. In general, MORGA variants are actually "better" than random search in all the problem instances but P5 and P9. The latter assumption is corroborated by the Wilcoxon text (Table 5), as the differences obtained by MACS and MORGA with respect to the multiobjective random search algorithm are both significant. The same stands for the unary metrics (Table 6) since the random search is clearly outperformed in the 10 instances by both MACS and MORGA according to the *HVR* values, and by MACS according to the *GD* values. In this latter metric, the random search is a little bit more competitive with MORGA since the former outperforms the latter in four instances.

NSGA-II vs. MACS 0.2 no-heur. The *C* metric results (boxplots in Fig. 7) show that NSGA-II "outperforms" MACS no-heur in five problem instances while the MACS scheme is "better" in other four. A similar performance is also noticed with the *GD* metric (Table 6). NSGA-II achieves better results in four instances and MACS in the other six. In view of the results of these two metrics, we could conclude that both algorithms behave in a similar way.

However, we should notice that this NSGA-II behaviour is somehow deceitful. The non-constructive but global search nature of NSGA-II and the problems derived from the use of an order encoding (see Section 2.5) cause a convergence of the generated Pareto fronts to a narrow region located in the left-most zone of the objective space (i.e. solutions with small values of *m*). Therefore, it lacks of an appropriate diversity to generate an extensive Pareto front in order to provide useful solutions to the problem being tackled, see the very bad values in *HVR* (Table 6), as well as the Pareto fronts in Figs. 8 and 9. Considering, for example, the P9 instance (Fig. 9), it can be seen that NSGA-II only reaches one non-dominated solution, although it belongs to the pseudo-optimal Pareto set. Note that, these are not satisfactory outcomes for the TSALBP-1/3 problem since they do not provide the decision maker with a number of good quality assembly line design choices presenting a different trade-off between the number of stations and the area of those stations. On the other hand, it generates extreme line configurations with a very small number of stations and a large area which, although valid as any other Pareto set solution, may be dangerous from an industrial point of view (the same as configurations with a very large number of stations and a small area, see Section 2.2).

Thus, this undesirable behaviour of the algorithm prompts very bad results in *HVR* and in the number of solutions of the Pareto front. However, NSGA-II achieves fairly good *C* and *GD* metric values since every solution it generates usually belongs to the true Pareto front. In addition, MACS no-heur achieves a significant difference in the dominance probability with respect to NSGA-II (see Table 5). This is due to the fact that as a consequence of the special Pareto front shapes generated

Table 6
Unary metrics for the 10 problem instances comparing MACS and MORGA with the random algorithm and NSGA-II.

	# dif_sols	*HVR*	*GD*	# dif_sols	*HVR*	*GD*
	P1			*P2*		
A1	10.9 (1.3)	0.16 (0.01)	383.95 (110.13)	11 (1.95)	0.40 (0.03)	562.4 (415.1)
A2	2.9 (0.94)	0.85 (0.03)	341.1 (277.01)	1.7 (0.64)	0.82 (0.03)	**318.2 (240.3)**
A3	10.7 (1.55)	**0.88 (0.01)**	**329.9 (51.8)**	**12.5 (1.5)**	**0.89 (0.01)**	401.5 (38.6)
A4	**11.5 (1.8)**	0.85 (0.01)	647.69 (44.14)	12.1 (1.37)	**0.89 (0.01)**	485.44 (451.32)
	P3			*P4*		
A1	9.8 (1.72)	0.22 (0.01)	15.07 (3.35)	**9.6 (1.11)**	0.52 (0.05)	45.84 (17.59)
A2	2.6 (1.11)	0.78 (0.08)	**4.77 (1.82)**	1 (0)	0.17 (0.08)	40.50 (27.59)
A3	7.1 (0.54)	0.55 (0.01)	24.11 (1.69)	9.3 (1.62)	0.84 (0.01)	135.62 (21.52)
A4	**12.8 (2.79)**	**0.88 (0.01)**	6.36 (1.12)	11 (0.89)	**0.92 (0.02)**	**14.15 (1.45)**
	P5			*P6*		
A1	**10.2 (0.98)**	0.91 (0.02)	7.57 (5.22)	6.3 (0.78)	0.20 (0.03)	4.71 (0.36)
A2	2 (0)	0.45 (0.01)	8.10 (3.02)	1.4 (0.49)	0.01 (0.01)	3.02 (0.19)
A3	6.7 (0.64)	0.90 (0.05)	21.79 (17.21)	**7.4 (0.8)**	**0.86 (0.04)**	1.65 (0.58)
A4	7.1 (0.3)	**0.94 (0)**	**3.58 (1.06)**	6.7 (0.64)	0.85 (0.02)	**1.15 (0.23)**
	P7			*P8*		
A1	8.1 (1.51)	0.35 (0.14)	13.78 (5.13)	10 (1.61)	0.42 (0.02)	19.69 (6.62)
A2	2.1 (0.3)	0.68 (0.04)	4.61 (1.82)	1.3 (0.46)	0.51 (0.06)	23.31 (10.84)
A3	7.2 (1.47)	0.62 (0.06)	8.34 (1.32)	**13.2 (0.98)**	**0.90 (0.01)**	**5.57 (0.59)**
A4	**10.3 (1.79)**	**0.73 (0.03)**	**5.30 (0.39)**	12.4 (1.36)	0.86 (0.01)	9.60 (1.44)
	P9			*P10*		
A1	9.4 (1.56)	0.53 (0.02)	71.33 (42.59)	**8.2 (0.98)**	0.62 (0.01)	6.11 (0.53)
A2	1 (0)	0.27 (0)	**0.30 (0.48)**	1.8 (0.6)	0.46 (0.21)	3.21 (1.61)
A3	3.9 (0.7)	0.81 (0.01)	697.17 (122.7)	6.7 (0.9)	0.82 (0.02)	2.91 (0.65)
A4	**14.6 (2.01)**	**0.89 (0.01)**	40.59 (7.53)	**8.2 (1.54)**	**0.87 (0.01)**	**2.38 (0.4)**

A1: Random search, A2: NSGA-II, A3: MORGA 0.3, A4: MACS 0.2 (no-heur).

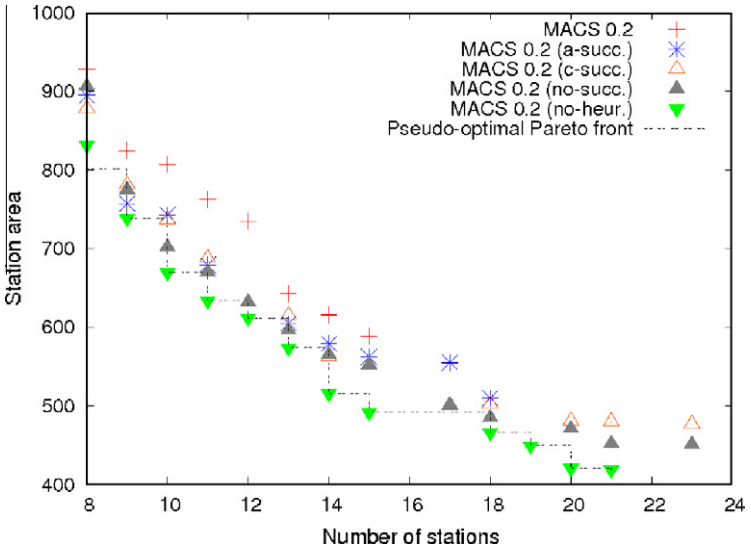

Fig. 8. Pareto fronts for the P4 instance.

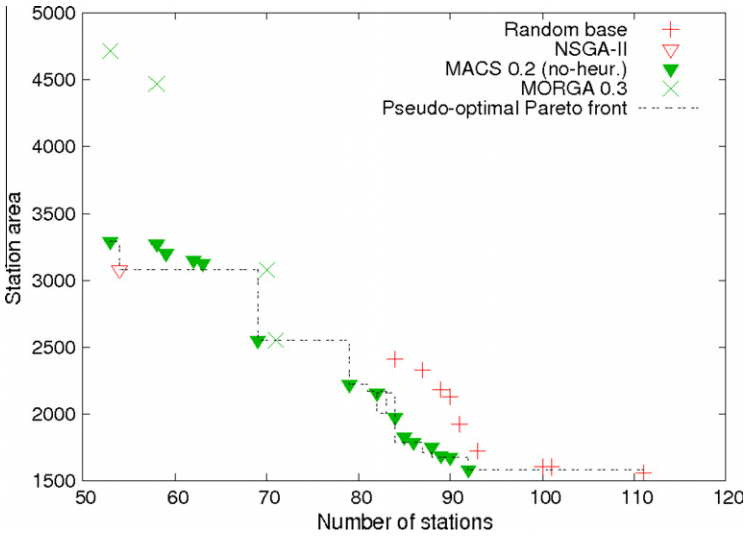

Fig. 9. Pareto fronts for the P9 instance.

by NSGA-II, the *C* metric always shows intermediate or low values (lower than 0.5 in many cases). On the contrary, the well spread Pareto fronts of MACS completely dominate the NSGA-II's ones in several cases. Note that, while the averaged dominance probability of MACS with respect to NSGA-II is 0.26, its counterpart is only 0.01. Thus, the *HVR* metric and the provided statistical test correct the *a priori* analysis from the *C* and *GD* metrics.

NSGA-II vs. MORGA 0.3. NSGA-II attains better *C* and *GD* and worse *HVR* metric results than MORGA 0.3. While NSGA-II outperforms MORGA in six instances concerning *GD*, the latter clearly outperforms the former in eight instances according to the *HVR* values. Conclusions are similar to those presented in the previous NSGA-II vs. MACS 0.2 no-heur analysis. However, since MORGA shows worse performance than MACS, there is no significant difference in the statistical analysis between MORGA and NSGA-II (Table 5).

Global conclusions. Overall, the main idea we conclude from these results is the good performance of MACS without heuristic information, which shows significant differences with respect to the multiobjective random search algorithm, MORGA, and NSGA-II.

Despite these MACS no-heur good results, it is important to remark that every pseudo-optimal Pareto set includes solutions that MACS no-heur was not able to obtain. For example, the NSGA-II approach, which is not able to properly spread the

Fig. 10. The real engine of Nissan Pathfinder. It consists of 747 pieces and 330 parts.

Pareto front, generally obtains a couple of non-dominated left-most solutions belonging to the pseudo-optimal Pareto set which are sometimes not achieved by MACS 0.2 no-heur (see Figs. 8 and 9).

4.5. A real-world case: Nissan Pathfinder engine

In this section we consider the application of the best algorithms designed to a real-world problem corresponding to the assembly process of the Nissan Pathfinder engine (shown in Fig. 10) at the plant of Barcelona (Spain). The assembly of these

Table 7
Mean and standard deviation $\bar{x}(\sigma)$ of the C metric values for the Nissan real-world problem.

	NSGA-II	MORGA 0.3	MACS 0.2	MACS 0.2 (no-heur)
NSGA-II	•	0.13 (0.01)	0.13 (0.02)	0.25 (0.05)
MORGA 0.3	0.05 (0.16)	•	0.51 (0.12)	0.36 (0.1)
MACS 0.2	0.05 (0.16)	0.92 (0.11)	•	0.44 (0.1)
MACS 0.2 (no-heur)	0.1 (0.32)	0.87 (0.1)	0.82 (0.1)	•

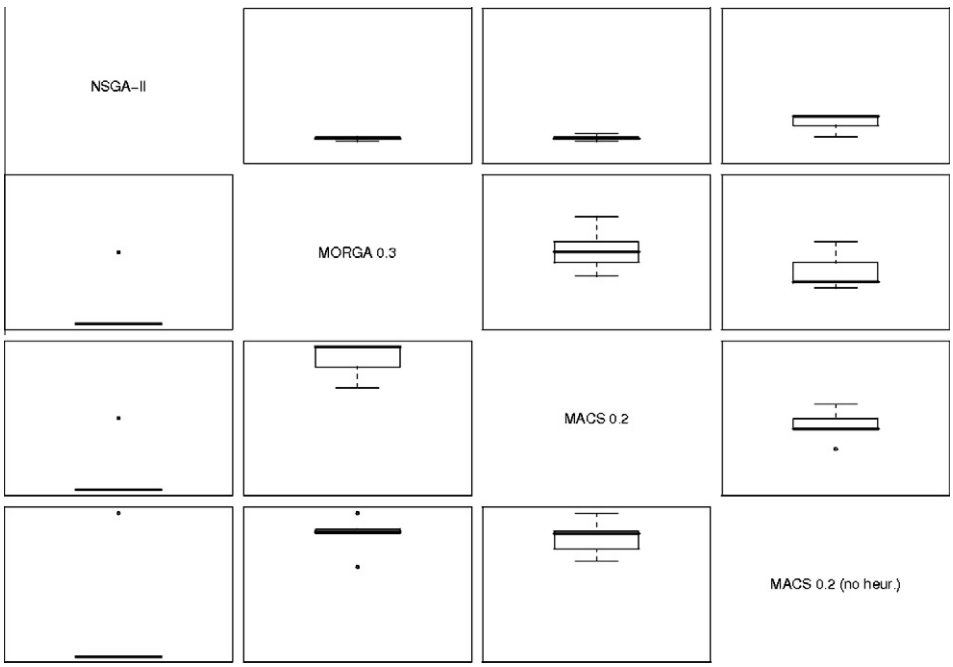

Fig. 11. C metric values represented by means of boxplots for the real-world problem instance of the Nissan Pathfinder engine.

Table 8
Averaged dominance probability and statistical significance for the Nissan real-world problem.

	NSGA-II	MORGA 0.3	MACS 0.2	MACS 0.2 (no-heur)
NSGA-II	•	0 =	0 =	0 =
MORGA 0.3	0 =	•	0.1 −	0 −
MACS 0.2	0 =	0.9 +	•	0 −
MACS 0.2 (no-heur)	0.1 =	0.9 +	0.8 +	•

Table 9
Mean and standard deviation $\bar{x}(\sigma)$ of the unary metric values for the Nissan real-world problem instance.

Method	Nissan with cycle time = 180		
	# dif_sols	*GD*	*HVR*
NSGA-II	1.2 (0.4)	**0.05** (0.11)	0.3446 (0.03)
MORGA 0.3	**7.6** (0.66)	1.13 (0.22)	0.8758 (0.01)
MACS 0.2	**7.6** (0.92)	1.12 (0.23)	0.8999 (0.01)
MACS 0.2 (no-heur)	**7.6** (1.02)	0.88 (0.17)	**0.9258 (0.01)**

engines is divided into 378 operation tasks, although we have grouped these operations into 140 different tasks. For more details about the Nissan instance the reader is referred to [5], where all the tasks and the time and area information are set.

From all the algorithms implemented, we have selected MORGA 0.3, MACS 0.2, and MACS 0.2 no-heur as well as the NSGA-II approach to tackle this problem instance. The *C* metric mean and standard deviation values are collected in Table 7. They are also represented by means of boxplots in Fig. 11. For a better comparison, Table 8 provides the results of the Wilcoxon statistical test. Besides, those values for the *HVR*, *GD* and the number of different solutions generated are shown in Table 9.

We can observe that MACS 0.2 no-heur is the "best algorithm" considering almost all the metrics. With respect to the *C* metric, the solutions generated by both MACS 0.2 versions dominate almost all MORGA solutions. As expected, MACS 0.2 no-heur attains better solutions than MACS 0.2 according to that metric (Table 7 and Fig. 11) and to the visualisation of the Pareto fronts in Fig. 12. Furthermore, MACS 0.2 no-heur is significantly "better" than the former two algorithms according to the dominance probability (Table 8). The analysis of NSGA-II shows the same conclusions than in the previous section. Its Pareto fronts are quite poor in terms of diversity and extension, although the only two pseudo-optimal Pareto solutions composing

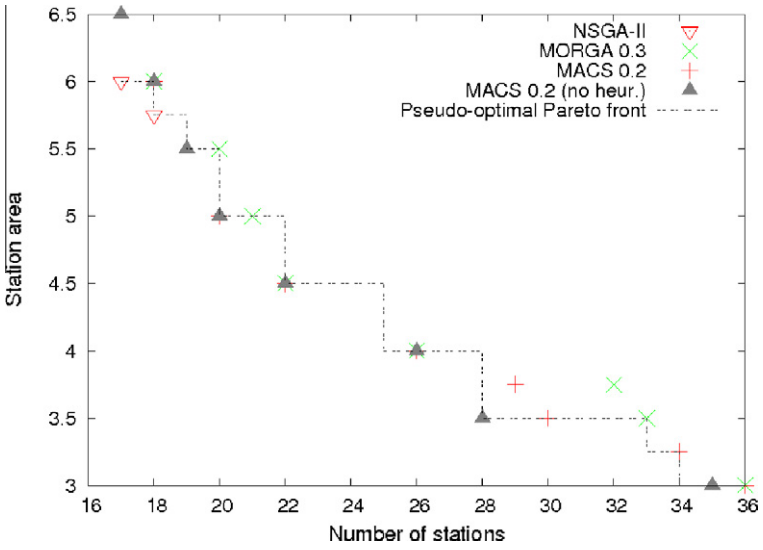

Fig. 12. Pareto fronts for the real-world problem instance of Nissan.

the aggregated Pareto front (see Fig. 12) are not obtained by the rest of algorithms. There is no significant difference among NSGA-II and the remainder according to the Wilcoxon test.

In Table 9, the values of the *HVR* metric show a good convergence of MACS no-heur. MORGA 0.3 and MACS 0.2 obtain a similar behaviour in terms of convergence to the pseudo-optimal Pareto front, with slightly better values for the latter, but at the cost of having a larger deviation. The bad *HVR* value of NSGA-II and the low number of found solutions can be also observed, although it shows the best *GD* value.

To sum up, and happened with the other problem instances, MACS no-heur outperforms NSGA-II, MORGA 0.3, and MACS 0.2 considering globally all the metrics. The statistical test for the dominance probability and the graphical Pareto front representation also validate this conclusion. The MACS algorithm is, in general, more suitable for the Nissan problem instance than NSGA-II and MORGA.

5. Concluding remarks

We have proposed new multiobjective constructive approaches to tackle the TSALBP-1/3. The performance of two solution procedures based on the MACS and MORGA algorithms with different design configurations have been presented and analysed. A multiobjective random search algorithm and a NSGA-II implementation were considered as baselines. Bi-objective variants of 10 assembly line problem instances have been used in the study as well as a real problem from a Nissan industrial plant.

From the obtained results we have concluded that the best yield to solve the problem globally corresponds to the MACS algorithm. Moreover, the use of a variant without heuristic information has reached even better results for most of the problem instances tackled, including the Nissan one. These conclusions were confirmed using a Wilcoxon test to analyse the statistical significance of the dominance probability of the algorithms. When we compared the results of all the MACS and MORGA runs, we noticed that both algorithms work better when we use 0.2 as a value for the q_0 parameter in the MACS transition rule, and 0.3 as γ control parameter in the MORGA RCL. Therefore, it is proven there is a need for increasing the diversity to obtain better results.

Several ideas for future developments arise from this work: (i) due to the features of our constructive procedures we can apply a local search to increase the performance of the algorithms, (ii) the merge of different search behaviours in just one multi-colony algorithm could be useful because of the impossibility of reaching the whole true Pareto front surface by a single algorithm, (iii) the consideration of other MOACO algorithms like P-ACO [21] or BicriterionMC [36] to solve the problem can be used to check if a different search behaviour allows us to improve the results, and (iv) the inclusion of user preferences to guide the multiobjective search process in the direction of the expert needs could be used [14,13].

Acknowledgements

This work has been supported by the UPC Nissan Chair and the Spanish Ministerio de Educacin y Ciencia under the PROTHIUS-II project (DPI2007-63026) and by the Spanish Ministerio de Ciencia e Innovacin under project TIN2009-07727, both including EDRF fundings.

References

[1] E.J. Anderson, M.C. Ferris, Genetic algorithms for combinatorial optimization: the assembly line balancing problem, ORSA Journal on Computing 6 (1994) 161–173.
[2] D. Angus, C. Woodward, Multiple objective ant colony optimization, Swarm Intelligence 3 (2009) 69–85.
[3] 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.
[4] J. Bautista, J. Pereira, Ant algorithms for assembly line balancing, Lecture Notes in Computer Science 2463 (2002) 67–75.
[5] 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.
[6] I. Baybars, A survey of exact algorithms for the simple assembly line balancing problem, Management Science 32 (8) (1986) 909–932.
[7] C. Blum, Beam-ACO for simple assembly line balancing, INFORMS Journal on Computing 20 (4) (2008) 618–627.
[8] C. Blum, J. Bautista, J. Pereira, Beam-ACO applied to assembly line balancing, Lecture Notes in Computer Science 4150 (2006) 96–107.
[9] J. Casillas, O. Cordón, I. Fernández de Viana, F. Herrera, Learning cooperative linguistic fuzzy rules using the best–worst ant systems algorithm, International Journal of Intelligent Systems 20 (2005) 433–452.
[10] V. Chankong, Y.Y. Haimes, Multiobjective Decision Making Theory and Methodology, North-Holland, 1983.
[11] W.C. Chiang, The application of a tabu search metaheuristic to the assembly line balancing problem, Annals of Operations Research 77 (1998) 209–227.
[12] M. Chica, O. Cordón, S. Damas, J. Bautista, Adding diversity to a multiobjective ant colony algorithm for time and space assembly line balancing, in: 2009 IEEE International Symposium on Assembly and Manufacturing (ISAM 2009), Suwon (Korea), 2009, pp. 364–379.
[13] M. Chica, O. Cordón, S. Damas, J. Bautista, Integration of an emo-based preference elicitation scheme into a multi-objective aco algorithm for time and space assembly line balancing, in: 2009 IEEE International Conference on Multicriteria Decision Making (IEEE MCDM 2009), Nashville (USA), 2009, pp. 157–162.
[14] M. Chica, O. Cordón, S. Damas, J. Bautista, J. Pereira, Incorporating preferences to a multi-objective ant algorithm for time and space assembly line balancing, in: Ant Colony Optimization and Swarm Intelligence, Lectures Notes in Computer Sciences, vol. 5217, Springer, 2008, pp. 331–338.
[15] M. Chica, O. Cordón, S. Damas, J. Bautista, J. Pereira, A multiobjective ant colony optimization algorithm for the 1/3 variant of the time and space assembly line balancing problem, in: 12th International Conference on Processing and Management of Uncertainty in Knowledge-based Systems (IPMU), Málaga (Spain), 2008, pp. 1454–1461.
[16] S.C. Chu, J.F. Roddick, J.S. Pan, Ant colony system with communication strategies, Information Sciences 167 (1–4) (2004) 63–76.
[17] C.A. Coello, G.B. Lamont, D.A. Van Veldhuizen, Evolutionary Algorithms for Solving Multi-objective Problems, second ed., Springer, 2007.

[18] O. Cordón, F. Herrera, T. Stützle, A review on the ant colony optimization metaheuristic: basis, models and new trends, Mathware and Soft Computing 9 (2–3) (2002) 141–175.
[19] K. Deb, Multi-objective Optimization Using Evolutionary Algorithms, Wiley, 2001.
[20] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197.
[21] K. Doerner, J.W. Gutjahr, R.F. Hartl, C. Strauss, C. Stummer, Pareto ant colony optimization: a metaheuristic approach to multiobjective portfolio selection, Annals of Operations Research 131 (1–4) (2004) 79–99.
[22] M. Dorigo, G. Di Caro, L.M. Gambardella, Ant algorithms for discrete optimization, Artificial Life 5 (2) (1999) 137–172.
[23] M. Dorigo, L. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 53–66.
[24] 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.
[25] M. Dorigo, T. Stützle, Ant Colony Optimization, MIT Press, Cambridge, 2004.
[26] K.A. Dowsland, B. William, Packing problems, European Journal of Operational Research 56 (1) (1992) 2–14.
[27] I. Ellabib, P.H. Calamai, O.A. Basir, Exchange strategies for multiple ant colony system, Information Sciences 177 (5) (2007) 1248–1264.
[28] T.A. Feo, M.G.C. Resende, Greedy randomized adaptive search procedures, Journal of Global Optimization 6 (1995) 109–133.
[29] L. Gambardella, E. Taillard, G. Agazzi, MACS-VRPTW: a multiple ant colony system for vehicle routing problems with time windows, in: News Ideas in Optimization, McGraw-Hill, London, 1999, pp. 63–76.
[30] 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 of study on the CEC'2005 Special Session on Real Parameter Optimization, Journal of Heuristics 15 (2009) 617–644.
[31] C. García Martínez, O. 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 (2007) 116–148.
[32] F. Glover, G.A. Kochenberger (Eds.), Handbook of Metaheuristics, Kluwer Academic, 2003.
[33] J.F. Gonçalves, J.R. Almeida, A hybrid genetic algorithm for assembly line balancing, Journal of Heuristics 8 (2002) 629–642.
[34] A. Heinrici, A comparison between simulated annealing and tabu search with an example from the production planning, in: H. Dyckhoff, U. Derigs, M. Salomon (Eds.), Operations Research Proceedings 1993, Berlin, Germany, 1994, pp. 498–503.
[35] T.R. Hoffmann, Assembly line balancing with a precedence matrix, Management Science 9 (1963) 551–562.
[36] S. Iredi, D. Merkle, M. Middendorf, Bi-criterion optimization with multi-colony ant algorithms, in: First International Conference on Evolutionary Multi-criterion Optimization (EMO'01), Springer, 2001, pp. 359–372.
[37] H. Ishibuchi, K. Narukawa, N. Tsukamoto, Y. Nojima, An empirical study on similarity-based mating for evolutionary multiobjective combinatorial optimization, European Journal of Operational Research 188 (1) (2008) 57–75.
[38] Y.K. Kim, Y. Kim, Y.J. Kim, Two-sided assembly line balancing: a genetic algorithm approach, Production Planning and Control 11 (2000) 44–53.
[39] J. Knowles, D. Corne, Instance generators and test suites for the multiobjective quadratic assignment problem, in: Proceedings of the 2002 Congress on Evolutionary Computation (CEC), vol. 1, Honolulu, HI, USA, 2002, pp. 711–716.
[40] Z.J. Lee, C.Y. Lee, A hybrid search algorithm with heuristics for resource allocation problem, Information Sciences 173 (1–3) (2005) 155–167.
[41] Y.Y. Leu, L.A. Matheson, L.P. Rees, Assembly line balancing using genetic algorithms with heuristic-generated initial populations and multiple evaluation criteria, Decision Sciences 25 (1994) 581–606.
[42] M. Middendorf, F. Reischle, H. Schmeck, Multi-colony ant algorithms, Journal of Heuristics 8 (3) (2002) 305–320.
[43] A.M. Robertson, P. Willett, Generation of equifrequent groups of words using a genetic algorithm, Journal of Documentation 50 (3) (1994) 213–232.
[44] I. Sabuncuoglu, E. Erel, M. Tayner, Assembly line balancing using genetic algorithms, Journal of Intelligent Manufacturing 11 (2000) 295–310.
[45] L. Sánchez, J.R. Villar, Obtaining transparent models of chaotic systems with multi-objective simulated annealing algorithms, Information Sciences 178 (2008) 950–970.
[46] A. Scholl, Balancing and Sequencing of Assembly Lines, second ed., Physica-Verlag, Heidelberg, 1999.
[47] A. Scholl, C. Becker, State-of-the-art exact and heuristic solution procedures for simple assembly line balancing, European Journal of Operational Research 168 (3) (2006) 666–693.
[48] G. Suresh, S. Sahu, Stochastic assembly line balancing using simulated annealing, International Journal of Production Research 32 (1994) 1801–1810.
[49] F.B. Talbot, J.H. Patterson, W.V. Gehrlein, A comparative evaluation of heuristic line balancing techniques, Management Science 32 (4) (1986) 430–455.
[50] C.F. Tsai, C.W. Tsai, C.C. Tseng, A new hybrid heuristic approach for solving large traveling salesman problem, Information Sciences 166 (1–4) (2004) 67–81.
[51] E. Zitzler, K. Deb, L. Thiele, Comparison of multiobjective evolutionary algorithms: empirical results, Evolutionary Computation 8 (2) (2000) 173–195.
[52] E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, V. Grunert da Fonseca, Performance assessment of multiobjective optimizers: an analysis and review, IEEE Transactions on Evolutionary Computation 7 (2) (2003) 117–132.