Engineering Applications of Artificial Intelligence 25 (2012) 254–273

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

Multiobjective memetic algorithms for time and space assembly line balancing

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

[a] European Centre for Soft Computing, 33600 Mieres, Spain
[b] Department of Computer Science and Artificial Intelligence, E.T.S. Informática y Telecomunicación, 18071 Granada, Spain
[c] Nissan Chair ETSEIB Universitat Politècnica de Catalunya, 08028 Barcelona, Spain

ARTICLE INFO

Available online 1 June 2011

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

ABSTRACT

This paper presents three proposals of multiobjective memetic algorithms to solve a more realistic extension of a classical industrial problem: time and space assembly line balancing. These three proposals are, respectively, based on evolutionary computation, ant colony optimisation, and greedy randomised search procedure. Different variants of these memetic algorithms have been developed and compared in order to determine the most suitable intensification–diversification trade-off for the memetic search process. Once a preliminary study on nine well-known problem instances is accomplished with a very good performance, the proposed memetic algorithms are applied considering real-world data from a Nissan plant in Barcelona (Spain). Outstanding approximations to the pseudo-optimal non-dominated solution set were achieved for this industrial case study.

1. Introduction

Nowadays, assembly lines are crucial in the industrial production of high quantity standardized commodities and more recently even gained importance in low volume production of customised products (Boysen et al., 2008). An assembly line is made up of a number of workstations, arranged either in series or in parallel. Since the manufacturing of a production item is divided into a set of tasks, a usual and difficult problem is to determine how these tasks can be assigned to the stations fulfilling certain restrictions. Consequently, the aim is to get an optimal assignment of subsets of tasks to the stations of the plant. Moreover, each task requires an operation time for its execution.

A family of academic problems – referred to as simple assembly line balancing problems (SALBP) – was proposed to model this situation (Baybars, 1986; Scholl, 1999). Taking this family as a base, Bautista proposed a more realistic framework: the time and space assembly line balancing problem (TSALBP) (Bautista and Pereira, 2007). The new model considers an additional space constraint to become a simplified version of real-world problems. As described in Bautista and Pereira (2007), this space constraint emerged due to the study of the Nissan plant in Barcelona, Spain (a snapshot of an assembly line of this industrial plant is shown in Fig. 1). The new TSALBP framework is of a great importance in industrial engineering and operations research since it achieves a better modelling of the real conditions of the balancing of assembly lines. The proposal of more realistic ALB models, allowing us to properly cope with real-life scenarios, have become a hot topic in the area in the last few years (Boysen et al., 2008).

As many real-world problems, TSALBP formulations have a multi-criteria nature (Chankong and Haimes, 1983) because they contain three conflicting objectives to be minimised: the cycle time of the assembly line, the number of the stations, and the area of these stations. In this paper we deal with the TSALBP-1/3 variant which tries to jointly minimise two objectives, the number of stations and their area, for a given value of the remaining objective, the product cycle time. TSALBP-1/3 has an important set of hard constraints-like precedences or cycle time limits for each station that make the problem solving difficult. These characteristics initially demanded the use of constructive approaches like ant colony optimisation (ACO) (Dorigo and Stützle, 2004) or greedy randomised search procedures (GRASP) (Feo and Resende, 1995) as done in the proposals described in Chica et al. (2010a,b), respectively. Nevertheless, an advanced proposal based on the well-known NSGA-II multiobjective evolutionary algorithm (Deb et al., 2002) has been recently introduced in Chica et al. (2011a) using a specific representation scheme and customised genetic operators for the TSALBP. The latter *advanced TSALBP-NSGA-II* proposal has overcome the problem shortcomings requiring a constructive technique and has outperformed the existing algorithms, becoming the state-of-the-art method.

* 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).

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

Memetic algorithms (MAs) (Moscato, 1989; Ong et al., 2006, 2010) have been widely used in industrial and engineering applications like the fleet vehicle routing problem (Prins, 2009), the design of spread spectrum radar poly-phase codes (Pérez-Bellido et al., 2008), the design of logistic networks (Pishvaee et al., 2010), or the construction of three-dimensional models of real-world objects (Santamaría et al., 2009). However, the use of local search to improve the solutions obtained by a global search procedure for the TSALBP has not been extensively explored (Bautista and Pereira, 2007; Chica et al., 2010b). In this paper, we aim to make an advance in the solving of this complex and challenging real-world problem by considering the application of advanced MA designs to deal with it.

We will design new multiobjective memetic methods for tackling the real-world TSALBP-1/3 variant. Such methods are based both on the state-of-the-art multiobjective algorithm, the *advanced TSALBP-NSGA-II*, and on the other existing multiobjective algorithms for the TSALBP. The new memetic proposals will incorporate a successful multi-criteria local search (LS) scheme used in a previous GRASP approach.

We aim at comparing different MA variants to show that there is no general method that is able to achieve the best results for all the problem instances (as stated in the *No Free Lunch* theorem Wolpert and Macready, 1997). Thus, we will develop 15 different MA designs to be compared to each other and to the basic global search methods in a complete experimentation with nine well-known problem instances.

Finally, an industrial case study will be considered to investigate the appropriateness of our MA proposals for solving real-world problems. This case study includes real-world data from the Nissan Pathfinder engine manufacturing process obtained from the assembly line of Barcelona. Up-to-date multiobjective performance indicators and statistical tests are used to analyse the behaviour of the algorithms.

The paper is structured as follows. In Section 2, the TSALBP-1/3 formulation is explained. The proposed multiobjective memetic algorithms to solve the problem are described in Section 3. Then, the experimental setup, the analysis of results, and the Nissan case study are presented in Section 4. Finally, some concluding remarks are discussed in Section 5.

2. Time and space assembly line balancing

The 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 as the workload of this station. Each task j can only be assigned to a single station k.

Every task j has a set of immediate "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. Thus, task j cannot be assigned to a station that is ordered 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. The workload time of the station cannot exceed the cycle time c, common to all the stations of the assembly line. In general, the SALBP (Baybars, 1986; Scholl, 1999) focuses on grouping these tasks into workstations by an efficient and coherent method. In short, the goal is to achieve a grouping of tasks that minimises the inefficiency of the line or its total downtime satisfying all the constraints imposed on the tasks and stations.

Nevertheless, this formulation is too simple to deal with real-life ALB problems. Different extensions to this formulation have been proposed (Scholl, 1999), showing the great interest of the scientific community (Boysen et al., 2008). In particular, there is a significant and real need of introducing space constraints in the assembly lines' design. This is because of three main reasons found in real manufacturing scenarios:

(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. This fact 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.

Based on these new realistic spatial features, a new real-like problem comes up. In order to model it, Bautista extended the SALBP into the TSALBP by means of the following formulation (Bautista and Pereira, 2007): the area constraint must be considered by associating a required area to each task. The areas of tasks are devoted to store auxiliary elements for manufacturing purposes like tools, shelves, containers, or hardware brackets. The needed area for each task is defined by the logistics and methods departments based on the characteristics of the involved auxiliary elements. We should keep in mind that the inclusion of space constraints in the problem formulation can decrease the efficiency with respect to a formulation which does not consider spatial constraints. However, these efficiency values only represent a theoretical nature because if spatial constraints are not included, the assembly line cannot be arranged.

Mainly, the required areas can be specified by two-dimensional units, i.e. length (a_j) and width (b_j). The first dimension, a_j, is the truly useful variable for the TSALBP optimisation task. From now on, the length associated to the tasks and the station's length will be referred as *area* and measured in linear metres. 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. This needed 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 $A = \max_{k=1,2,\ldots,m} A_k$. This fact is not problematic since if there is a continuous transportation system, as in our case, the areas of the stations must be equal. Otherwise, the velocity of the conveyor belt would require to be changed at each station and adapted to the cycle time. A diagram with an example is given in Fig. 2 where the area A_k of station k is given by the sum of the areas (lengths) of its tasks, a_1, a_2, a_3, and a_4.

Overall, 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 such 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 (c, the cycle time; m, the number of stations; and A, the area of the stations) are tackled as objectives to be optimised and which will be considered as fixed variables. The eight possible combinations result in eight different TSALBP variants (Bautista and Pereira, 2007). 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 three-objective problem.

In this contribution we will tackle one of these formulations, the TSALBP-1/3. It consists of minimising the number of stations m and the station area A, given a fixed value of the cycle time c. We chose this variant because it is quite realistic in the automotive industry, our field of interest, since the annual production of an industrial plant (and, therefore, the cycle time c) is usually set by market objectives. Besides, the search for the best number of stations and areas makes sense if we want to reduce costs and make workers' day better by setting up less crowded stations. More information about the justification of this choice can be found in Chica et al. (2010a).

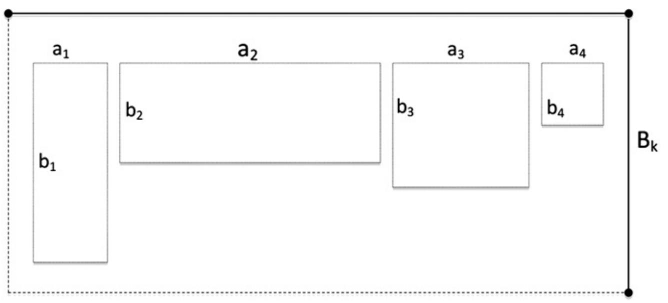


Fig. 2. A diagram showing the area configuration of a station k containing 4 different tasks. The important space dimension for the optimisation problem is the length of the tasks, a_i, that is called area in this paper.

We can mathematically formulate the TSALBP-1/3 variant as follows:

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

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

subject to:

$$\sum_{k=E_j}^{L_j} x_{jk} = 1, \quad j = 1,2,\ldots,n, \tag{3}$$

$$\sum_{k=1}^{UB_m} \max_{j=1,2,\ldots,n} x_{jk} \le m, \tag{4}$$

$$\sum_{j=1}^{n} t_j x_{jk} \le c, \quad k = 1,2,\ldots,UB_m, \tag{5}$$

$$\sum_{j=1}^{n} a_j x_{jk} \le A, \quad k = 1,2,\ldots,UB_m, \tag{6}$$

$$\sum_{k=E_i}^{L_i} k x_{ik} \le \sum_{k=E_j}^{L_j} k x_{jk}, \quad j = 1,2,\ldots,n; \;\; \forall i \in P_j, \tag{7}$$

$$x_{jk} \in \{0,1\}, \quad j = 1,2,\ldots,n; \;\; k = 1,2,\ldots,UB_m, \tag{8}$$

where:

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

Constraint in Eq. (3) restricts the assignment of every task to just one station, (4) limits decision variables to the total number of stations, (5) and (6) are concerned with time and area upper bounds, (7) denotes the precedence relationship among tasks, and (8) expresses the binary nature of variables x_{jk}.

The specialised literature includes a large variety of exact and heuristic problem-solving procedures as well as metaheuristics for solving the SALBP (Baybars, 1986; Scholl and Voss, 1996; Scholl and Becker, 2006). Regarding the TSALBP-1/3, a multiobjective ACO algorithm based on the multiple ant colony system (MACS) (Barán and Schaerer, 2003) was the first successful proposal (Chica et al., 2010a). However, a later multiobjective evolutionary algorithm, the *advanced TSALBP-NSGA-II*, outperformed MACS and became the state-of-the-art method (Chica et al., 2011a). Procedures based on other metaheuristics as GRASP have also been proposed (Chica et al., 2010b). Finally, expert preferences were modelled and included into the metaheuristic search process (Chica et al., 2011b, 2008).

The said three approaches to tackle TSALBP-1/3 will be described in Section 3.1 as the global search modules of our memetic proposals. With respect to the use of MAs, an ACO algorithm incorporating an LS strategy was proposed in Bautista and Pereira (2007) to solve a single-objective TSALBP variant. Nevertheless, no multiobjective MA design has been proposed to deal with any multiobjective TSALBP variant. The current contribution aims at bridging this gap.

3. Proposed memetic algorithms

In this section we introduce different advanced MA designs for tackling our industrial problem. Generally, (multiobjective) MAs may be regarded as a marriage between a (multiobjective) global search metaheuristic and local improvement operators. This general structure has actually proved its efficacy when solving a large number of real-world problems. Unfortunately, it is well known that there is not any universal MA design to deal with any general application. In fact, one drawback of MAs is that, in order for it to be useful, their general structure must be adapted to cope with the characteristics of the individual search components considered and of the problem under solving. These elements and how they are integrated to obtain the best performance are the pieces of the MA puzzle. Designers must use their knowledge, skills and expertise to make decisions on the composition of each individual procedure and of their integration in order to reach the best possible MA structure for the specific application being tackled (Ishibuchi et al., 2003; Ong et al., 2006, 2010). Some tentative designs based on the analysis of several combinations with a different intensification–diversification trade-off must be tested to succeed in this task.

The latter design process is a consequence of the fact that each (multiobjective) global search metaheuristic has its own peculiarities and defines different intensification–diversification degrees when combined with a LS method. Therefore, it is necessary to detail each global search method and how all the components are integrated in the final scheme for each specific MA case. As an example, in the design of a multiobjective MA for the current problem we found that the three different multiobjective metaheuristics to be considered as global search methods handle the final set of solutions, i.e. Pareto-optimal solutions, in different ways. On the one hand, these solutions can be stored in an external Pareto archive, as in MACS and GRASP. On the other hand, they can be included in the general population of the metaheuristic, as in the advanced TSALBP-NSGA-II. These specific decisions are those not allowing for a universal MA design.

The structure of the current section keeps these ideas in mind and follows the usual MA design pipeline. To do so, in Section 3.1 the three basic multiobjective global search methods tested are reviewed. Then, the LS structure and operators are introduced in Section 3.2. Finally, Section 3.3 describes the different chances considered for the LS integration within the global search scheme.

3.1. Global search: multiobjective metaheuristics

We describe the three multiobjective metaheuristic designs which have been applied to the TSALBP-1/3, i.e. the MACS algorithm, a GRASP method, and the state-of-the-art *advanced TSALBP-NSGA-II*.

3.1.1. MACS

MACS (Barán and Schaerer, 2003) was proposed as an extension of ant colony system (ACS) (Dorigo and Gambardella, 1997) to deal with multiobjective problems. In Chica et al. (2010a), the authors modified the original version of MACS to adapt it for solving the TSALBP-1/3. The algorithm uses one pheromone trail matrix and several heuristic information functions. In the case of the TSALBP-1/3, the experimentation carried out in Chica et al. (2010a) showed that the performance was better when MACS was only guided by the pheromone trail information. Therefore, the heuristic information functions have not been considered in this contribution.

Since the number of stations is not fixed, the method is based on constructive and station-oriented approach (Scholl, 1999) to face the precedence problem (as usually done for the SALBP, Scholl and Becker, 2006). Thus, the algorithm opens a station and sequentially selects tasks to fill it by means of the MACS transition rule till a stopping criterion is reached. Then, a new station is opened to be filled and the procedure is iterated till all the existing tasks are allocated.

The pheromone information has to memorise which tasks are the most appropriate to be assigned to a station. Hence, a pheromone trail has to be associated to a pair ($station_k$,$task_j$), $k=1\ldots n$, $j=1\ldots n$, with n being the number of tasks, so the pheromone trail matrix has a bi-dimensional nature. Since MACS is Pareto-based, i.e. a set of non-dominated solutions for the problem is stored in a Pareto archive and updated at each step of the algorithm, the pheromone trails are updated using the solutions of this archive. Two station-oriented single-objective greedy algorithms are used to obtain the initial pheromone value τ_0.

In addition, a novel mechanism was introduced in the construction procedure in order to achieve a better search intensification–diversification trade-off. This mechanism randomly decides when to close the current station taking as a base both a station closing probability distribution and an ant filling threshold $\alpha_i \in [0,1]$. The probability distribution is defined by the station filling rate (i.e. the overall processing time of the current set of tasks S_k assigned to that station) as follows:

$$p(closing\ k) = \frac{\sum_{i \in S_k} t_i}{c}. \tag{9}$$

At each construction step, the current station filling rate is computed. In case it is lower than the ant's filling percentage threshold α_i (i.e. when it is lower than $\alpha_i \cdot c$), the station is kept opened. Otherwise, the station closing probability distribution is updated and a random number is uniformly generated in [0,1] to take the decision whether the station is closed or not. If the decision is to close the station, a new station is created to allocate the remaining tasks. Otherwise, the station will be kept open. Once the latter decision has been taken, the next task is chosen among all the candidate tasks using the MACS transition rule to be assigned to the current station as usual:

$$j = \begin{cases} \underset{j \in \Omega}{\operatorname{argmax}}(\tau_{ij} \cdot [\eta^0_{ij}]^{\lambda\beta} \cdot [\eta^1_{ij}]^{(1-\lambda)\beta}), & \text{if } q \le q_0, \\ \hat{i}, & \text{otherwise.} \end{cases} \tag{10}$$

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 $\hat{i}$ is selected according to the probability distribution $p(j)$ of Eq. (11). This probability is applied to perform a controlled exploration of the neighbourhood Ω at each decision node of the ant, as done in the original ACS. Again, β weights the relative importance of the heuristic information with respect to the pheromone trails and λ depends on each ant index

$$p(j) = \begin{cases} \dfrac{\tau_{ij} \cdot [\eta^0_{ij}]^{\lambda\beta} \cdot [\eta^1_{ij}]^{(1-\lambda)\beta}}{\sum_{u \in \Omega} \tau_{iu} \cdot [\eta^0_{iu}]^{\lambda\beta} \cdot [\eta^1_{iu}]^{(1-\lambda)\beta}}, & \text{if } j \in \Omega, \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

The procedure goes on till there are no remaining tasks to be assigned. Thus, the higher the ant's threshold, the higher the probability of a totally filled station, and *vice versa*. 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 method to properly explore the different parts of the optimal Pareto front by appropriately distributing the generated solutions.

The algorithm performs a local pheromone update every time an ant crosses an edge $\langle i,j \rangle$ using the average costs of the τ_0 value. It is done as follows:

$$\tau_{ij} = (1-\rho)\cdot\tau_{ij} + \rho\cdot\tau_0 \tag{12}$$

The interested reader is referred to Chica et al. (2010a) for a complete description of the MACS proposal for the TSALBP-1/3.

3.1.2. GRASP

Another successful metaheuristic applied to the TSALBP-1/3 was a multiobjective GRASP method[1] (Chica et al., 2010b). With this approach, a solution is generated at each iteration and its inclusion in the external Pareto archive is considered: if it is not dominated, it is included in the archive and the resulting dominated solutions are removed. The algorithm finishes with a set of non-dominated solutions generated during all the iterations.

As in MACS, the construction method is based on a station-oriented approach. In the construction of the greedy solutions we introduce randomness in two processes. On the one hand, we allow the random selection of the next task among the best candidates to be assigned to the current station. This process starts by creating a candidate list of unassigned tasks. For each candidate task j, we compute its heuristic value η_j. It measures the preference of assigning it to the current opened station. η_j is proportional to the processing time and area ratio of that task (normalised with the upper bounds given by the time cycle, c, and the sum of all tasks' areas, UB_A, respectively). In addition, η_j is also proportional to the ratio between the number of successors of task j and the maximum number of immediate successors of any eligible task:

$$\eta_j = \frac{t_j}{c}\cdot\frac{a_j}{UB_A}\cdot\frac{|F_j|}{\max_{i\in\Omega}|F_i|} \tag{13}$$

Then, we sort all the candidate tasks according to their heuristic values and we set a quality threshold for them given by $q = \max_{\eta_j} - \gamma\cdot(\max_{\eta_j} - \min_{\eta_j})$. All the candidate tasks with a heuristic value η_j greater or equal to q are selected to be in the restricted candidate list (RCL). In the former expression, γ is the intensification–diversification trade-off control parameter. We found that $\gamma = 0.3$ was the value that yield the best performance (Chica et al., 2010a). Finally at the current construction step, we randomly select a task among the elements of the RCL. The construction procedure finishes when all the tasks have been allocated in the needed stations.

On the other hand, we also introduce randomness in the decision of closing the current station according to a probability distribution given by the filling rate of the station (see Eq. (9)). As stated in MACS, the filling thresholds approach is also used to achieve a diverse enough Pareto front. A different threshold is selected in isolation at each iteration of the multiobjective randomised greedy algorithm, i.e. the construction procedure of each solution considers a different threshold.

The algorithm is run a number of iterations to generate different solutions. When a solution is generated a local improvement phase is performed on the solution. This improvement is achieved by means of a multi-criteria LS scheme, later explained in Section 3.3. The final output consists of a Pareto set approximation composed of the non-dominated solutions found.

[1] Unlike MACS and NSGA-II, a GRASP approach always includes a LS improvement applied to the constructed solutions. Therefore, we will not consider the constructive step without the local improvement in this work.

3.1.3. Advanced TSALBP-NSGA-II

In Chica et al. (2011a) the authors proposed a novel multi-objective genetic algorithm design, called *advanced TSALBP-NSGA-II*, and based on the original NSGA-II search scheme (Deb et al., 2002). Customised representation and operators were considered in the algorithm design to properly solve the TSALBP-1/3 by considering a global search technique.

The most important problem of the previous genetic algorithm-based approaches that tried to solve the SALBP and TSALBP (see for example Chica et al., 2010a and Sabuncuoglu et al., 2000) was the representation scheme. The *advanced TSALBP-NSGA-II* proposal took the biggest step ahead with respect to existing algorithms by explicitly representing task-station assignments regardless the cycle time of the assembly line. Thus, it ensures a proper search space exploration for the joint optimisation of the number and the area of the stations. Furthermore, the representation will also follow an order encoding to facilitate the construction of feasible solutions with respect to the precedence relations constraints. The allocation of tasks among stations is made by employing separators, that are dummy genes which do not represent any specific task and they are inserted into the list of genes representing tasks. In this way, they define groups of tasks being assigned to a specific station.

The maximum possible number of separators is $n-1$ (with n being the number of tasks), as it would correspond to an assembly line configuration with n stations. The number of separators included in the genotype is variable and it depends on the number of existing stations in the current solution. Therefore, the algorithm works with a variable-length coding scheme, although its order-based representation nature avoids the need of any additional mechanism to deal with this issue.

Due to the latter fact, the crossover operator can be designed from a classical order-based one. The partially mapped crossover (PMX) operator was selected because (a) it is one of the most extended crossover operators, and (b) it has already been used in other genetic algorithm implementations for the SALBP (Sabuncuoglu et al., 2000). PMX generates two offspring from two parents by means of the following procedure: (a) two random cut points are selected, (b) for the first offspring, the genes outside the random points are copied directly from the first parent, and (c) the genes inside the two cut points are copied but in the order they appear in the second parent. Thanks to the advanced coding scheme and to the use of a permutation-based crossover, the feasibility of the offspring with respect to precedence relations is assured.

However, since information about the tasks-stations assignment is encoded inside the chromosome, it is needed to assure that: (a) there is not any station exceeding the fixed cycle time limit, and (b) there is not any empty station in the configuration of the assembly line. Therefore, a repair operator must be applied for each offspring after crossover. The goals and methods of the repair operator are: (a) redistribute spare tasks among available stations by reallocating the spare tasks in other stations, and (b) removing empty stations.

Two mutation operators have also been specifically designed and uniformly applied to the selected individuals of the population. The first one, the *scramble operator*, is based on reordering a part of the sequence of tasks and reassigning them to stations. The second one, the *divider operator*, is introduced to induce more diversity in order to achieve a well-distributed Pareto front approximation.

In order to additionally increase the diversity of the search to obtain better distributed Pareto front approximations, a diversity induction mechanism was adopted: Ishibuchi et al.'s (2008) similarity-based mating.

The interested reader is referred to Chica et al. (2011a) for a complete description of the method.

3.2. Multi-criteria LS structure and components

Mainly, there are two stochastic LS approaches for multi-objective combinatorial optimisation problems (Teghem and Jaszkiewicz, 2003; Paquete and Stützle, 2006). The first one uses an acceptance criterion based on the weak component-wise ordering of the objective value vectors of neighbouring solutions. In addition, it maintains an unbounded archive of non-dominated solutions found during the search process (a Pareto archive) (Knowles and Corne, 2003; Zitzler and Thiele, 1999). The second family is based on considering different scalarizations of the objective function vector (Gandibleux and Freville, 2000; Hansen, 1997; Jaszkiewicz, 2002). The MA designs introduced in this contribution will be based on this second approach. The weighted sum scalarization of the two objectives of our problem, A and m, are calculated by the following formula:

$$\text{Min} \quad (\lambda^1 A+\lambda^2\, m). \tag{14}$$

This will be the function to be optimised by the multi-criteria LS of the MAs. As usually done in the multiobjective MA area (see for example Jaszkiewicz, 2002), the weight vector $\lambda=(\lambda^1,\lambda^2)$ is created at random for each constructed solution.

first, ES_j, and last station, LP_j, where task j may be re-assigned by the corresponding LS operator according to the current assignment of its immediate predecessors and successors. In general, a move (j,k_1,k_2) describes the assignment change of task j from station k_1 to station k_2, where $k1 \neq k2$ and $k2 \in [ES_j,LP_j]$. ES_j and LP_j are variables of the LS algorithm. They are re-calculated each time a LS operator is going to be applied by locating where the immediate precedent task of j, s, and the immediate successor of j, p, are placed in the existing solution. Note that they should not be confused with E_j and L_j which are definitions of the TSALBP model and restrict the set of stations where the corresponding task j could be never allocated (see Section 2).

The pseudo-code of the LS operator for the first objective, A, is described in Algorithm 1. In this method, the solution neighbourhood is built by means of the explained task moves. The main goal is to reduce the area occupied by the station with the highest area by moving tasks to other stations. It works by first sorting the tasks of a target station and selecting the task with the highest area. Then, the algorithm tries to move this task to one of its feasible stations in order to reduce the scalarization value of the solution. If there is no possible improvement with this task, the algorithm selects the next task of the sorted list of tasks of the target station.

Algorithm 1. The pseudo-code of the LS operator for the A objective.

```
1   while $Iterations \leq MAX\_ITERATIONS$ do
2       Target_Station ← Find the station with the highest area;
3       Tasks ← Descending_Sort(tasks of Target_Station);
4       while no scalarization function improvement AND Tasks $\neq \emptyset$ do
5           Task ← First element of Set_Of_Tasks;
6           Find $ES_j$ and $LP_j$ of Task $j$;
7           while no scalarization function improvement do
8               Possible_Station ← station with the lowest area
9                 $\in [ES_j,LP_j]$;
10              Move Task from Target_Station to Possible_Station;
11              if scalarization function improvement then
12                  Make the move permanent;
13              end
14          end
15          Remove Task from Set_Of_Tasks;
16      if $Target\_Station = \emptyset$ then
17          Remove Target_Station;
18      end
19      Iterations ← Iterations + 1;
20  end
21  return true if scalarization function is improved;
```

The existing local improvement procedures for ALB are based on moves (Rachamadugu and Talbot, 1991). The LS operators are based on such moves of tasks. In our advanced specific design, two different neighbour generation operators will be considered and selected depending on the weight vector λ (see Section 3.2). If $\lambda^1 > \lambda^2$, the neighbour operator for minimising the A objective will be followed since the LS optimisation will be more biased to the improvement of the latter objective than the other. Otherwise, the neighbour operator headed to improve m will be considered first. If the selected neighbour operator does not succeed minimising the weighted sum scalarization, the other operator is then applied.

To explain the operation mode of both operators it is necessary to define, for each task j in the current TSALBP-1/3 solution, the

In the case of the second LS operator, the goal is reducing the number of stations m. From the initial solution, a neighbourhood is created by moving all the tasks from the station with the lowest number of tasks (called the Target_Station) to other stations, keeping a feasible solution. The operator works as described in Algorithm 2. For a sorted list of stations with respect to the number of tasks, the algorithm tries to move all the tasks of each station in order to improve the scalarization function value. This is done for a maximum number of stations. Given a station to be removed, the algorithm uses a recursive depth first search function (Algorithm 3) to look for a feasible solution having the Target_Station's tasks reallocated in other stations. In the experiments developed, the maximum number of stations (*MAX_STATIONS*) was set to 20 to limit the computational time of this LS operator.

Algorithm 2. The pseudo-code of the LS operator for the *m* objective.

```
1  while $Iterations \le MAX\_ITERATIONS$ do
2      Set_Of_Stations ← Ascending Sort (with respect to no. of tasks);
3      $i \leftarrow 1$;
4      while $i \le MAX\_STATIONS$ AND no scalarization function improvement do
5          Target_Station ← i − th element of Set_Of_Stations;
6          Set_Of_Tasks ← Descending_Sort(tasks of Target_Station);
7          for all elements of Set_Of_Tasks do
8              Find $ES_j$ and $LP_j$;
9          end
10         First_Element = First Element of Set_Of_Tasks;
11         DFS(First_Element,Set_Of_Tasks);
12         if no scalarization function improvement then
13             $i \leftarrow i+1$;
14         end
15     end
16     Iterations ← Iterations + 1;
17 end
18 return true if scalarization function is improved;
```

Algorithm 3. The pseudo-code of the Depth First Search implemented in a recursive fashion, used by the LS operator for objective *m*.

```
1  Function DFS (Current_Task, Set_Of_Tasks)
2  if all elements of Set_Of_Tasks allocated then
3      // Base case
       Calculate scalarization function of the objective function vector;
4  else
5      for all the possible stations of Current_Task do
6          Move Current_Task to the selected station if feasible;
7          // Recursive call of the Depth First Search algorithm
8          Next_Task ← Next task of Set_Of_Tasks;
9          DFS(Next_Task,Set_Of_Tasks);
10         if no scalarization function improvement then
11             Undo Current_Task movement;
12         end
       end
13 end
14 return true if scalarization function is improved;
```

3.3. Multiobjective LS integration

The most important issue in the LS integration scheme in a MA is the balance between the application of the basic global search method and the LS (Ishibuchi et al., 2003). In memetic computing, LS is usually applied to each trial solution obtained during the global search process. However, this is very time-consuming process and it has been reported that this do not necessarily lead to the best performing MA (Krasnogor and Smith, 2000).

An alternative choice is considering a selective application of the LS as done for example in Ishibuchi et al. (2003), Herrera et al. (2005) and Noman and Iba (2005). That is one of the alternatives we will use in this work. We have considered a criterion that was originally proposed in Hart (1994) and later used in contributions such as Krasnogor and Smith (2000), Lozano et al. (2004) and Santamaría et al. (2009). It is based on a random application with uniform distribution considering a probability value of 0.0625. We will compare this criterion with the traditional scheme of applying the LS improvement to every constructed solution during the global search process.

Another issue that could significantly affect the MA intensification–diversification trade-off is the LS depth measured by the number of LS iterations we are considering. The higher the number of LS iterations, the higher the intensification (and the lower the diversification) the MA is applying. We will consider three different number of LS iterations, 20, 50, and 100, and study their influence in the experiments developed.

4. Experimentation

In this section we aim at studying the performance and behaviour of the different designed multiobjective MAs. First, we describe the experimental setup (Section 4.1). Then, an analysis of the MA variants performance is done (Section 4.2.2). Finally, the real-world case study of Nissan is tackled in Section 4.3.

4.1. Experimental setup

We run each algorithm 10 times with different random seeds, setting a fixed run time as stopping criterion (900 s). All the algorithms were launched in the same computer: Intel Pentium^{TM} D with two CPUs at 2.80 GHz and CentOS Linux 4.0 as operating system. Furthermore, the same programming language, C++, and framework were utilised for the development of all the algorithms here described. *The framework with the algorithms of the experimental study is publicly available at* http://www.nissanchair.com/TSALBP. The specific parameter values considered for the different algorithms are shown in Table 1.

We will consider the two usual kinds of multiobjective performance indicators (metrics) existing in the specialised literature (Zitzler et al., 2000, 2003; Deb, 2001; Knowles and Corne, 2002; Coello et al., 2007): (a) unary performance indicators, those which measure the quality of a non-dominated solution set approximation returned by an algorithm; and (b) binary performance indicators, those which compare the performance of two different multiobjective algorithms. In the following paragraphs we present a brief description of the used performance indicators:

Hypervolume ratio unary indicator: The hypervolume ratio (*HVR*) (Coello et al., 2007) has become a very useful unary performance indicator. Its use is very extended as it can jointly measure the distribution and convergence of a Pareto set approximation. The *HVR* can be calculated as follows:

$$HVR = \frac{HV(P)}{HV(P^*)}, \tag{15}$$

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

Since we are working with real-world problems we have to keep in mind some obstacles which make difficult the computation of this performance indicator. First, we should notice that the true Pareto fronts are not known. In our case, we will consider a pseudo-optimal Pareto set, i.e. an approximation of the true Pareto set, obtained by merging all the Pareto set approximations P_i^j generated for each problem instance by any algorithm in any run. Thanks to this pseudo-optimal Pareto set, we can compute the *HVR* performance indicator values, considering them in our analysis of results.

Besides, there is an additional problem with respect to the *HVR* performance indicator. In minimisation problems, as ours, there is a need to define a reference point to calculate the volume of a given Pareto front. The *HVR* values are not proper to be compared if there is not any upper boundary of the region within which all feasible points will lie (Knowles and Corne, 2002). Thus, we defined the reference point for each instance as the "logical" maximum values for the two objectives (anti-ideal solution). These reference points are specific for each problem instance.

I_ε binary performance indicator: The previous performance indicator allows us to determine the absolute and individual quality of a Pareto front approximation, but cannot be used for comparison purposes (Zitzler et al., 2003). On the opposite, binary indicators aim to compare the performance of two different multiobjective algorithms by comparing the Pareto set approximations generated by each of them. In this contribution, we will consider the ε binary indicator, I_ε.

The I_ε indicator (Zitzler et al., 2003) is a quality assessment method for multiobjective optimisation that avoids particular difficulties of unary and classical methods (Knowles, 2006). Two different definitions are possible: the standard (multiplicative) I_ε and the additive indicator $I_{\varepsilon+}$. In this contribution, we have opted by the multiplicative indicator. Given two Pareto front approximations, P and Q, the value $I_\varepsilon(P,Q)$ is calculated as follows:

$$I_\varepsilon(P,Q) = \inf_{\varepsilon \in \Re}\{\forall z^2 \in Q, \exists z^1 \in P : z^1 \preceq_\varepsilon z^2\}, \tag{16}$$

where $z^1 \preceq_\varepsilon z^2$ iff $z_i^1 \le \varepsilon \cdot z_i^2$, $\forall i \in \{1,\ldots,o\}$, with o being the number of objectives, assuming minimisation.

According to Zitzler et al. (2003), the I_ε binary indicator can be properly used to compare the performance of two different multiobjective algorithms by analysing the crossed values of the metric as follows. If both $I_\varepsilon(P,Q) \le 1$ and $I_\varepsilon(Q,P) > 1$, then it can be considered that the Pareto set approximation P generated by the first algorithm dominates Q, the one generated by the second algorithm, in a weak sense.

The I_ε performance indicator values of the approximation sets of the 10 runs performed for every pair of algorithms have been represented by two kinds of boxplots (see Figs. 3, 5, and 7; and Figs. 9, 13, 14, respectively). For all the boxplots, 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, a thick line within the box shows the median, and the isolated points are the outliers of the distribution.

In the first kind of boxplots (Figs. 3, 5, and 7), each rectangle contains nine boxplots representing the distribution of the I_ε values for a certain ordered pair of algorithms in the nine considered problem instances (see Section 4.2.1). Each box refers to the algorithm A in the corresponding row and algorithm B in

Table 1
Used parameter values for the multiobjective MAs.

Parameter	Value	Parameter	Value
MACS			
Number of ants	10	β	2
ρ	0.2	q_0	0.2
Ants' thresholds (2 ants per each)	{0.2, 0.4, 0.6, 0.7, 0.9}		
GRASP			
γ	0.3	Diversity thresholds	{0.2, 0.4, 0.6, 0.7, 0.9}
Advanced TSALBP-NSGA-II			
Population size	100	Ishibuchi's similarity-based mating γ, δ values	10
Crossover probability	0.8	Mutation probability	0.1
α values for scramble mutation	{0, 0.8}		
LS			
Application criteria	{always, selective}	No. of iterations	{20, 50, 100}

the corresponding column, and gives the $I_\varepsilon(A,B)$ values. The 10 considered values to obtain each boxplot correspond to the computation of the I_ε metric on the two Pareto sets generated by algorithms A and B in each of the 10 runs.

The second kind of boxplots (Figs. 9, 13, 14) facilitates the analysis when few algorithms are involved in the comparison. In this case, each rectangle represents one of the nine problem instances. Inside each rectangle, boxplots representing the distribution of the I_ε values for a certain pair of algorithms are drawn. Given Fig. 9 as an example, the top-left rectangle shows the boxplots comparing three pairs of algorithms: M vs. G, M vs. N, and G vs. N (see the caption of the figure for the notations) for the first problem instance. As I_ε is a binary indicator, two boxplots have been drawn for each algorithm comparison. The white boxplots represent the distributions $I_\varepsilon(M,G)$, $I_\varepsilon(M,N)$, and $I_\varepsilon(G,N)$ generated in the 10 runs, while the coloured boxplots do so for the $I_\varepsilon(G,M)$, $I_\varepsilon(N,M)$, and $I_\varepsilon(N,G)$ values.

In order to allow an easy visual comparison of the performance of the different algorithms, the attainment surfaces (Fonseca and Fleming, 1996) will be represented. These graphics offer a visual and quantitative information, sometimes more useful than numeric values, mainly in complex problems as ours. We can define an attainment surface as the surface uniquely determined by a set of non-dominated points that divides the objective space into the region dominated by the set and the region that is not dominated by it (Fonseca and Fleming, 1996). Given r runs of an algorithm, it would be interesting to summarise the r attainment surfaces obtained, using only one summary surface. Such summary attainment surfaces can be defined by imagining a diagonal line in the direction of increasing objective values cutting through the r attainment surfaces generated. The intersection on this line that weakly dominates at least $r-p+1$ of the surfaces and is weakly dominated by at least p of them, defines one point on the "p-th summary attainment surface". In our case, this surface is the union of all the goals that been attained in the $r=10$ independent runs of the algorithm.

Finally, a statistical test will be performed in order to analyse the significance of the results in the comparison of the quality of the Pareto front approximations obtained by the different multiobjective MAs by means of the I_ε indicator. This is done in order to avoid the fact that one exceptionally good result in any of the repetitions of the compared algorithms could be responsible for the differences in the overall values and results in a wrong analysis. The Mann–Whitney U test, also known as Wilcoxon ranksum test, will be used for this aim. 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 (García et al., 2009).

Nevertheless, we should remark the fact that there is not any reference methodology to apply a statistical test to a binary indicator in multiobjective optimisation. Thus, we have decided to follow the procedure proposed in Sánchez and Villar (2008), described as follows. 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 approximation P generated by A dominates Q obtained by B, 0 otherwise. For comparisons with the I_ε indicator, it is considered that the Pareto set approximation P dominates Q when $I_\varepsilon(P,Q) \leq 1$ and $I_\varepsilon(Q,P) > 1$, as stated in Zitzler et al. (2003). Given 10 repetitions $B_1,\ldots,B_{10}$ of the multiobjective algorithm B, let $P_A(B) = (1/10)\sum_{i=1}^{10} p_A(B_i)$. Given another 10 repetitions $A_1,\ldots,A_{10}$ of A, let $P_A(B) = (P_{A_1}(B), P_{A_2}(B), \ldots, P_{A_{10}}(B))$. The vector $P_A(B)$ can be seen as a sample of a random variable with an overall number of 100 different observations representing the fraction of times that the output of algorithm A dominates that of algorithm B. If the expectation of $P_A(B)$ is greater than the expectation of $P_B(A)$, then we can state that algorithm A is better than algorithm B for the current experiment, since it is more likely that results of the former improve those of the latter than the opposite.

Hence, in order to know if there is a significant difference between the performance of the two compared algorithms, we can use a Wilcoxon test (null hypothesis $E(P_A(B)) = E(P_B(A))$, alternate hypothesis $E(P_A(B)) > E(P_B(A))$) to discard the expectations of the probability distributions $P_A(B)$ and $P_B(A)$ are the same. The significance level considered in all the tests to be presented is $p=0.05$. Besides, notice that, in case of including more than one problem instance in the comparison, as done in Section 4.2.3, $\widehat{P_A(B)}$ 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 all the considered problem instances.

4.2. Preliminary analysis on nine well-known problem instances

In this section we will show the results of the proposed MAs for nine different real-like problem instances. The analysis developed will serve us as a first step to apply the algorithms to the real-world problem instance in Section 4.3.

4.2.1. Problem instances

Nine problem instances with different features have been selected for this first experimentation: `arc111` with cycle time limits of $c=5755$ and 7520 (P1 and P2), `barthol2` (P3), `barthold` (P4), `lutz2` (P5), `lutz3` (P6), `mukherje` (P7), `scholl` (P8), and `weemag` (P9). They have been chosen to be as diverse as possible to test the performance of the algorithms and their variants when they deal with different problem conditions.[2] Originally, these instances were SALBP-1 instances[3] only having time information. However, we have created their area information by reverting the task graph to make them bi-objective (as done in Bautista and Pereira, 2007). The nine TSALBP-1/3 instances considered are publicly available at http://www.nissanchair.com/TSALBP.

4.2.2. Analysis of the results of the memetic approaches

We have run the different MA variants resulting from the use of the three different global search methods (i.e. MACS, GRASP, and TSALBP-NSGA-II), the two different LS application criteria (always or selective), and the three LS iterations number (20, 50, and 100). Therefore, we will have six memetic MACS variants, three memetic GRASP methods,[4] and six memetic variants of the *advanced TSALBP-NSGA-II*. All of them will also be benchmarked against the two basic global search approaches not considering the use of LS (i.e. MACS and TSALBP-NSGA-II).

Memetic MACS algorithm: We have designed three memetic variants with 20, 50, and 100 iterations applying the LS to all the solutions (MACS-LS1, MACS-LS2, and MACS-LS3, respectively), and other three variants with 20, 50, and 100 iterations but only applying randomly the LS to a 0.0625 percent of the generated solutions (MACS-LS4, MACS-LS5, and MACS-LS6, respectively). The *HVR* values are shown in the first 14 rows of Table 2. The boxplots of the I_ε performance indicator values of the memetic

[2] Note that 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.

[3] Available at http://www.assembly-line-balancing.de

[4] As said, a GRASP approach always include a local search improvement applied to every constructed solution. Hence, we just focus on the number of allowed iterations for the LS.

Table 2
Mean and standard deviation $\bar{x}(\sigma)$ of the *HVR* performance indicator values for the different variants of the MACS (M), GRASP (G), and *advanced TSALBP-NSGA-II* (TN) MAs. Higher values indicate better performance. Underlined values are the best results of each algorithm while bold values corresponds to the global best result.

Algorithm abbreviation	P1	P2	P3	P4	P5	P6	P7	P8	P9
	Memetic MACS algorithm								
M	0.7597 (0.004)	0.7581 (0.01)	0.6605 (0.009)	0.7129 (0.015)	0.5052 (0.014)	0.5744 (0.023)	0.7181 (0.01)	0.5081 (0.006)	0.7107 (0.008)
M-LS1	0.9463 (0.003)	0.9614 (0.003)	0.9154 (0.002)	0.9384 (0.015)	0.7440 (0.008)	0.8921 (0.015)	0.9834 (0.001)	0.8156 (0.002)	0.9053 (0.004)
M-LS2	0.9479 (0.003)	0.9643 (0.003)	0.9186 (0.003)	0.9535 (0.018)	0.7440 (0.008)	0.8921 (0.015)	**0.9888 (0.001)**	0.8316 (0.003)	0.9053 (0.004)
M-LS3	0.9479 (0.003)	0.9643 (0.003)	0.9186 (0.003)	0.9535 (0.018)	0.7440 (0.008)	0.8921 (0.015)	**0.9888 (0.001)**	0.8316 (0.003)	0.9053 (0.004)
M-LS4	0.9221 (0.006)	0.9439 (0.008)	0.8924 (0.004)	0.9516 (0.01)	0.6840 (0.013)	0.8216 (0.012)	0.9725 (0.001)	0.7969 (0.005)	0.8671 (0.011)
M-LS5	0.9267 (0.005)	0.9494 (0.007)	0.8975 (0.003)	0.9462 (0.013)	0.6848 (0.013)	0.8216 (0.012)	0.9773 (0.002)	0.8113 (0.005)	0.8671 (0.011)
M-LS6	0.9267 (0.005)	0.9494 (0.007)	0.8975 (0.003)	0.9462 (0.013)	0.6848 (0.013)	0.8216 (0.012)	0.9773 (0.002)	0.8112 (0.005)	0.8671 (0.011)
	GRASP								
G-LS1	0.9727 (0.001)	0.9646 (0.001)	0.8427 (0.002)	0.9758 (0.003)	0.7640 (0.009)	0.9146 (0.006)	0.9721 (0.002)	0.8065 (0.002)	0.9267 (0.003)
G-LS2	0.9750 (0.002)	0.9685 (0.001)	0.8483 (0.003)	0.9859 (0.002)	0.7640 (0.009)	0.9146 (0.006)	0.9773 (0.002)	0.8115 (0.003)	0.9267 (0.003)
G-LS3	0.9750 (0.002)	0.9683 (0.001)	0.8483 (0.003)	0.9853 (0.002)	0.7640 (0.009)	0.9146 (0.006)	0.9768 (0.002)	0.8115 (0.003)	0.9267 (0.003)
	Memetic advanced TSALBP-NSGA-II								
TN	0.9853 (0.004)	0.9474 (0.015)	0.8286 (0.049)	0.9411 (0.012)	0.7528 (0.047)	0.8962 (0.057)	0.8891 (0.022)	0.9346 (0.038)	0.8174 (0.015)
TN-LS1	**0.9953 (0.003)**	**0.9911 (0.003)**	**0.9819 (0.010)**	0.9908 (0.003)	**0.9444 (0.014)**	**0.9769 (0.012)**	0.9875 (0.003)	**0.9874 (0.007)**	0.9627 (0.013)
TN-LS2	0.9931 (0.005)	0.9907 (0.004)	0.9788 (0.009)	0.9986 (0.001)	0.9300 (0.023)	0.9767 (0.011)	0.9885 (0.003)	0.9490 (0.046)	**0.9647 (0.007)**
TN-LS3	0.9922 (0.005)	0.9904 (0.004)	0.9770 (0.008)	**0.9987 (0.001)**	0.9300 (0.023)	0.9767 (0.011)	0.9884 (0.003)	0.9486 (0.046)	**0.9647 (0.007)**
TN-LS4	0.9775 (0.012)	0.9710 (0.012)	0.9506 (0.009)	0.9906 (0.004)	0.8500 (0.047)	0.9374 (0.018)	0.9730 (0.003)	0.9314 (0.032)	0.9092 (0.014)
TN-LS5	0.9790 (0.008)	0.9676 (0.015)	0.9509 (0.007)	0.9979 (0.001)	0.8220 (0.04)	0.9331 (0.022)	0.9763 (0.004)	0.9340 (0.03)	0.9095 (0.014)
TN-LS6	0.9790 (0.008)	0.9676 (0.015)	0.9509 (0.007)	0.9983 (0.001)	0.8220 (0.04)	0.9331 (0.022)	0.9763 (0.004)	0.9341 (0.03)	0.9095 (0.014)

MACS variants are shown in Fig. 3. The analysis of the obtained results arises that:

- The basic MACS algorithm is clearly outperformed by every memetic MACS variant. The difference is significant in view of the *HVR* values in Table 2 and the I_ε boxplots in Fig. 3.
- The memetic MACS variants which applied the LS operator to all the generated solutions, i.e. MACS-LS1, MACS-LS2, and MACS-LS3, outperform those variants which selectively applied the LS operator (MACS-LS4, MACS-LS5, MACS-LS6) in every problem instance. Again, both performance indicators show the same conclusion.
- There is no difference in performance between running the LS operator with 50 and 100 iterations (MACS-LS2 and MACS-LS3, respectively). Therefore, the appropriate trade-off is obtained with 50 iterations and running the memetic MACS algorithm for more iterations is not necessary.
- The latter memetic variants, MACS-LS2 and MACS-LS3, are the best ones in view of the results obtained in both performance indicators. They show a better convergence than the memetic MACS-LS1 and, of course, than the memetic variants that applied less intensification in the LS operator (MACS-LS4, MACS-LS5, MACS-LS6). The latter facts are confirmed by the attainment surface plots of the Pareto front approximations generated by the memetic MACS variants in Fig. 4.
- However, in some instances, MACS-LS1 obtains solutions of the Pareto front that are not achieved by the “best” MACS-based MAs, MACS-LS2 and MACS-LS3. This situation can also

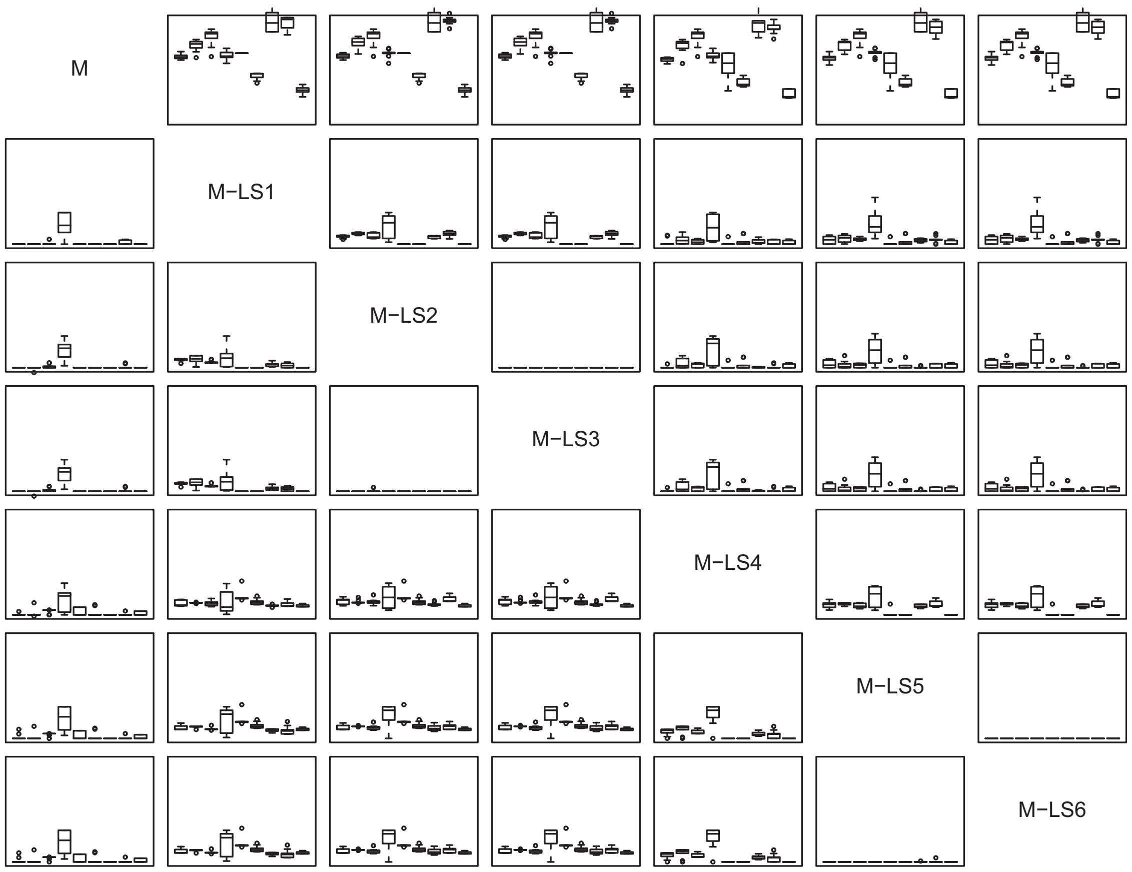


Fig. 3. I_ε values represented by means of boxplots comparing different memetic variants of the MACS algorithm.

be observed in the attainment surface plot at the bottom of Fig. 4. It is due to the fact that MACS-LS1 induces more diversity in the search process rather than a higher intensification by means of more LS iterations as applied by the other two variants.

GRASP: We analyse the behaviour of the GRASP methods with different LS intensification degrees. According to the *HVR* performance indicator values (central part of Table 2), the boxplots in Fig. 5 with I_ε values, and the attainment surface plots of Fig. 6, the most important considerations are:

- Overall, the variants with more LS iterations (GRASP-LS2 and GRASP-LS3) again outperform the variant with only 20 iterations (GRASP-LS1).
- There is a need of running the LS operators more than 20 iterations in all the problem instances but P5, P6, and P9 (see *HVR* values and I_ε boxplots).
- The best Pareto front approximations are obtained by the algorithms that apply the highest number of LS iterations (see Fig. 6).

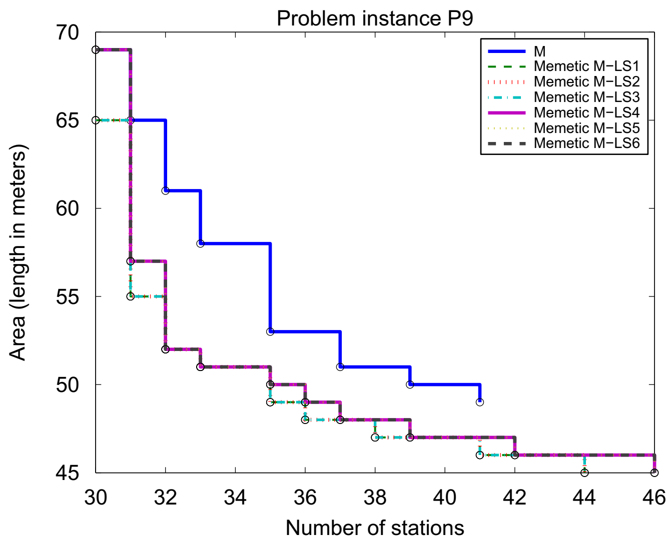
Memetic advanced TSALBP-NSGA-II : The *HVR* values corresponding to these memetic designs are collected at the bottom part of Table 2 while the corresponding values of the I_ε performance indicator are depicted in Fig. 7. In this case, we can conclude that:

- As in MACS, the MAs show a better performance than the basic *advanced TSALBP-NSGA-II*. However, in this case the difference between the MAs and the basic global search methods is lower because of the outstanding performance of the basic *advanced TSALBP-NSGA-II*.
- Applying the LS to all the solutions found by the *advanced TSALBP-NSGA-II* is again better than considering a selective application. We can observe how TN-LS1, TN-LS2, and TN-LS3 outperform the other three variants in both the *HVR* values of Table 2 and the I_ε boxplots of Fig. 7.
- Unlike the other two designs, i.e. memetic MACS and GRASP, the best memetic *advanced TSALBP-NSGA-II* is the TN-LS1 variant, which runs the LS operator just 20 iterations. Only in instances P4, P7, and P9, the memetic variants with higher LS intensification achieve better performance, but with a very low difference. The attainment surface plot in Fig. 8 corroborates this conclusion, showing how the use of less iterations (more diversification rather than intensification) allows obtaining some solutions that are not reached by the MAs that consider more LS iterations.

4.2.3. Global analysis

In this section we will summarise the global conclusions of the performance of the different memetic approaches proposed for solving the TSALBP-1/3:

- The application of the multi-criteria LS method to every solution generated by the global search methods is always better than using a selective criterion based on its application to the 0.0625% of those solutions.
- Normally, 50 iterations are enough for the LS methods. Therefore, spending time by running more iterations is not recommended since the obtained intensification–diversification trade-off performs equal or worst.
- In order to achieve the best solutions, a good exploratory global search method as the *advanced TSALBP-NSGA-II* is



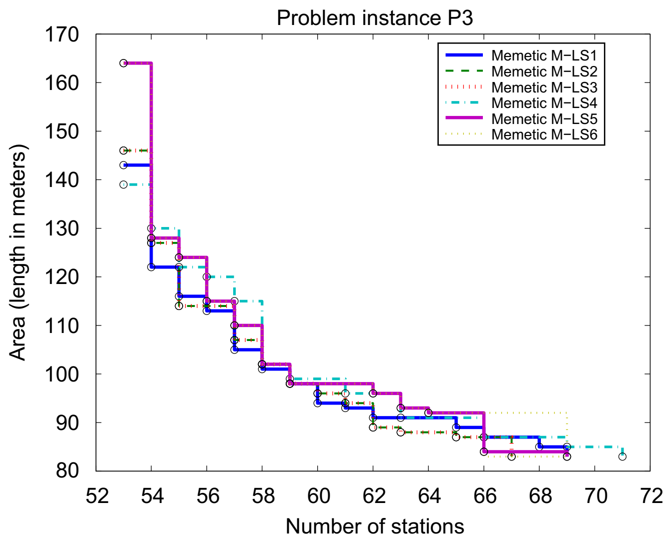


Fig. 4. Attainment surface plots of the MACS MAs for instances P3 and P9.

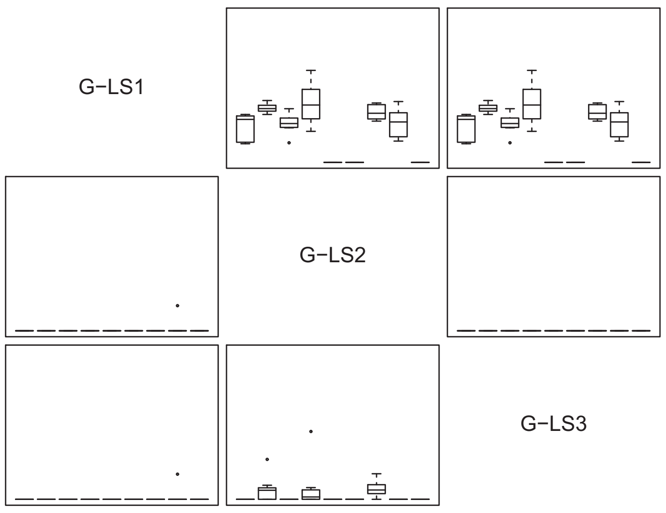


Fig. 5. I_ε values represent by means of boxplots comparing different GRASP variants.

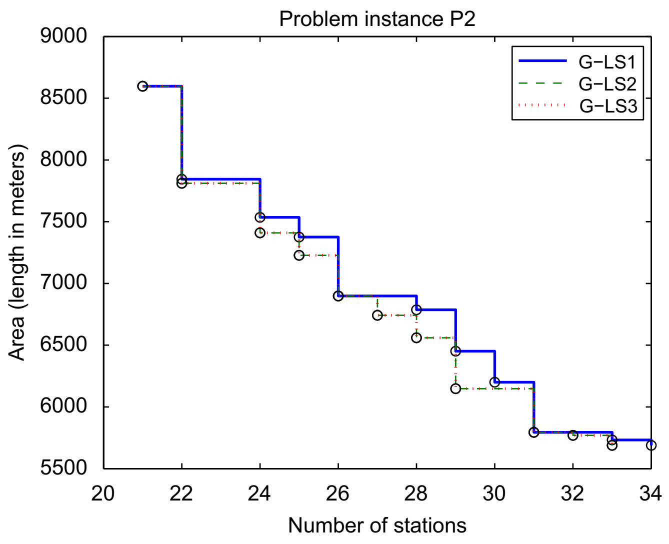


Fig. 6. Attainment surface plots of the GRASP methods for instance P2.

needed. If we apply the LS method to global search procedures that do not explore conveniently the search space, some regions of the Pareto front will never be reached. The use of the *advanced TSALBP-NSGA-II* allows its associated memetic design to spend less time in the LS intensification. This conclusion is drawn in view of the fact that the best MA in this group is the TN-LS1, then, TN-LS2 and TN-LS3, and finally, the rest of the memetic variants, TN-LS4, TN-LS5, and TN-LS6, which behave similarly.

- We can also provide a similar ranking of the memetic MACS algorithms: MACS-LS2, MACS-LS3, MACS-LS1, MACS-LS4, MACS-LS5, MACS-LS6. Nevertheless, some similar facts to those described in the previous item can be recognised in the MACS algorithm, where some solutions are only achieved by the MAs considering the lowest number of LS iterations.
- As expected, GRASP is the metaheuristic that performs a worst global search. It needs more LS iterations than the other MAs, probably because of the low quality of the solutions generated in the global search stage. GRASP-LS3, GRASP-LS2, and GRASP-LS1 are the MAs in order of performance.

By selecting the best variant of each memetic design, MACS-LS2, GRASP-LS3, and TN-LS1, it can be clearly observed how there is a strong relation between the quality of the global search and the number of iterations required by the LS. When we use worse global search procedures, more iterations in the LS provide better results. The selected best variants will be compared to each other but taking in mind that these best variants can change depending on the instance.

We have used the same performance indicators to reach the conclusions, i.e. the *HVR* values of Table 2, the I_ε boxplots of Fig. 9 comparing the three MAs, and the attainment surface plots (two of them are shown in Fig. 11). For a better comparison, a statistical test is also applied on the dominance probabilities calculated for the I_ε indicator on every pair of algorithms. These dominance probabilities are shown in the boxplots of Fig. 10. See Section 4.1 to recall their calculation process.

Table 3 provides the results of the Wilcoxon statistical test on the dominance probabilities of the best variants of the algorithms. Every cell of the table includes the *p*-values for the nine 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 performance (using the I_ε indicator) than the one that appears in the column.

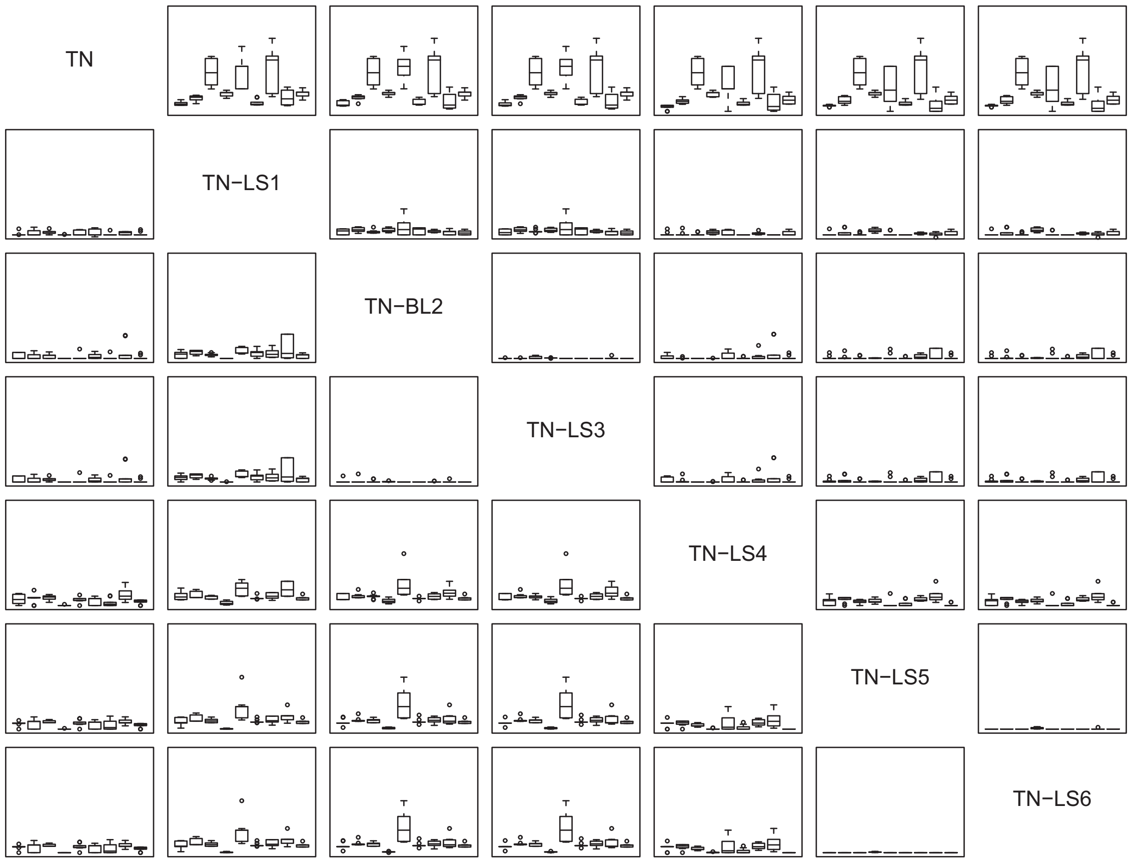


Fig. 7. I_ε values represented by means of boxplots comparing different memetic variants of the advanced TSALBP-NSGA-II.

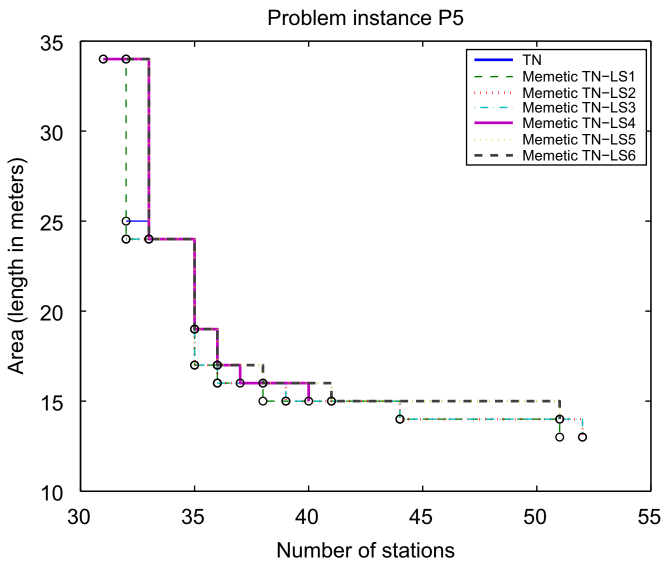


Fig. 8. Attainment surface plots of the memetic *advanced TSALBP-NSGA-II* for instance P5.

The clearest conclusion in view of the indicators is the memetic *advanced TSALBP-NSGA-II* is the best MA. It obtains better *HVR* and I_ε performance indicator values in all the problem instances but P7. This is the only problem instance where the *advanced TSALBP-NSGA-II* is not the best algorithm. In this case, the memetic MACS algorithm outperforms the remainder. Although there is not a big difference between the latter two algorithms, the memetic *advanced TSALBP-NSGA-II* is worse in P7 because of the performance variability of its runs. The memetic MACS algorithm is more stable and achieves similar behaviour in the 10 runs corresponding to the latter problem instance.

The good performance of the *advanced TSALBP-NSGA-II* is again clear looking at the dominance probabilities of Fig. 10 and the results of the statistical test shown in Table 3. In this analysis, the results obtained by the *advanced TSALBP-NSGA-II* are significantly better (represented by means of a "+" symbol in the table) than those by the rest of the algorithms, MACS and GRASP.

A comparison between the memetic MACS and GRASP is more difficult since their behaviour varies depending on the problem instance. The memetic MACS algorithm performance is better than GRASP in P3, P7, and P8, but worse in P1, P4, and P9. In P2, P5, and P6, they behave similarly and the values of the performance indicators are very close. The results of the Wilcoxon statistical test are in line with this analysis since there is no significance between them as can be observed from the "=" symbol of Table 3. Therefore, it cannot be stated which of these two MAs is the best one without focusing on a particular instance. The attainment surface plots in Fig. 11 corroborate this fact.

4.3. Experimentation on the Nissan case study

In the last section of the experimentation we will apply the proposed MAs to a real-world case study. First, we will describe the Nissan case study in Section 4.3.1 and then we will present and analyse the obtained results in Sections 4.3.2 and 4.3.3.

4.3.1. Nissan case study description

We consider the application of the best MA variants to a real-world problem corresponding to the assembly process of the

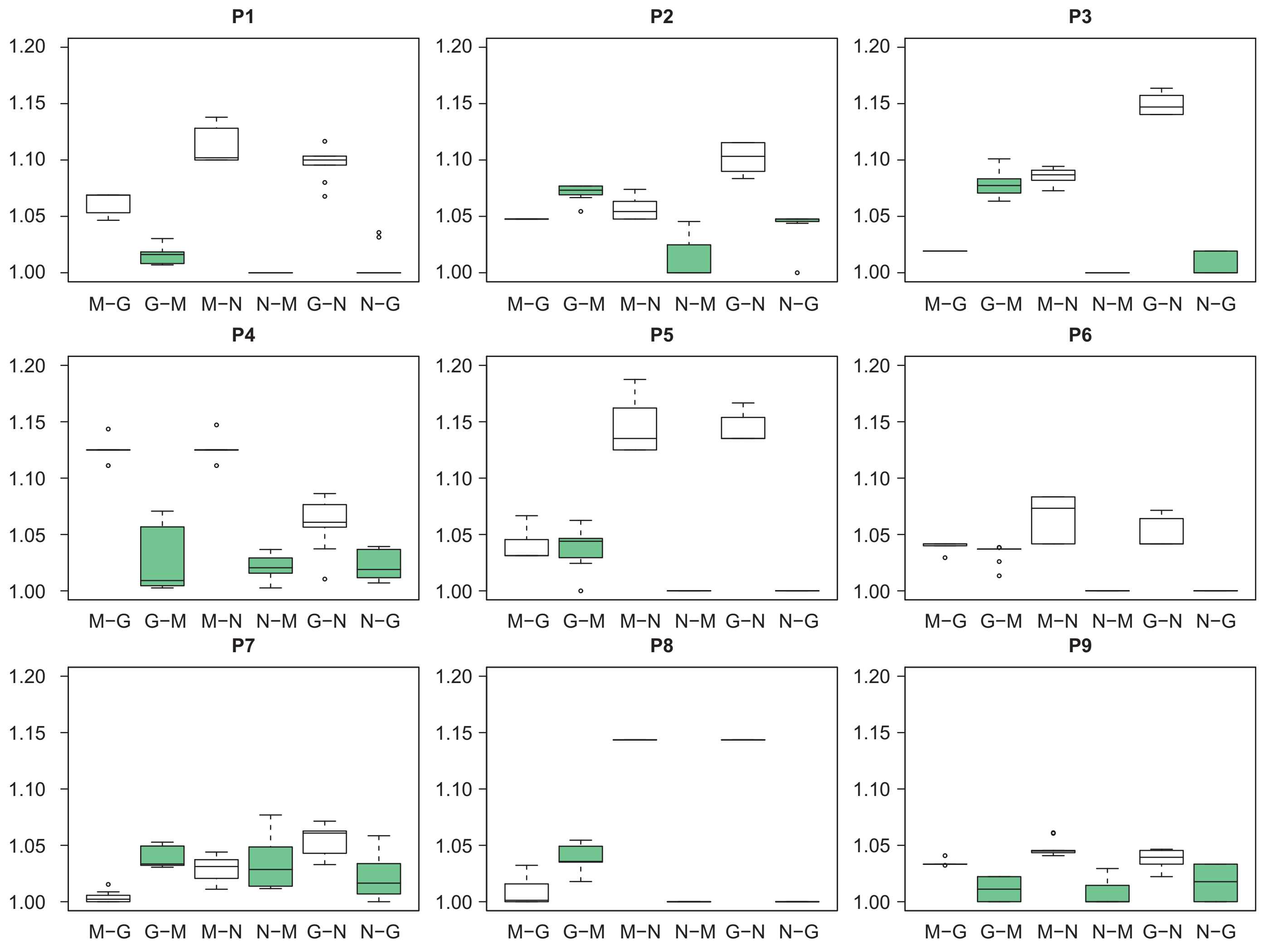


Fig. 9. I_ε boxplots comparing the best variant of each memetic design in the 9 instances (one rectangle per instance). The memetic MACS-LS2 is noted by M, GRASP-LS3 by G, and the *advanced TSALBP-NSGA-II-LS1* by N.

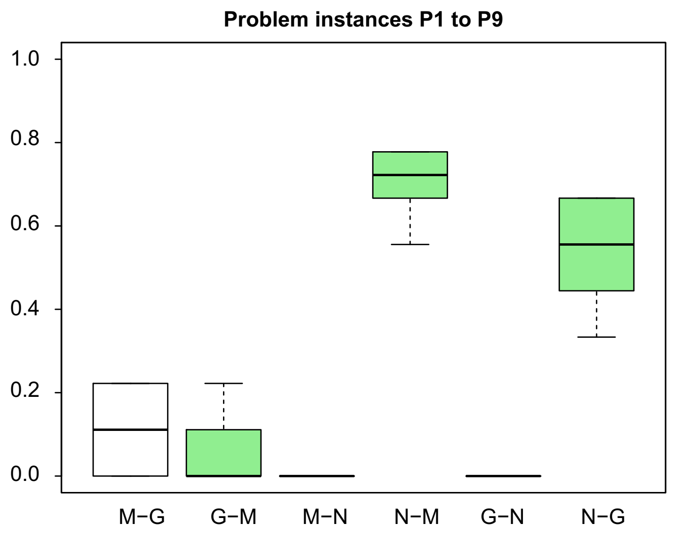
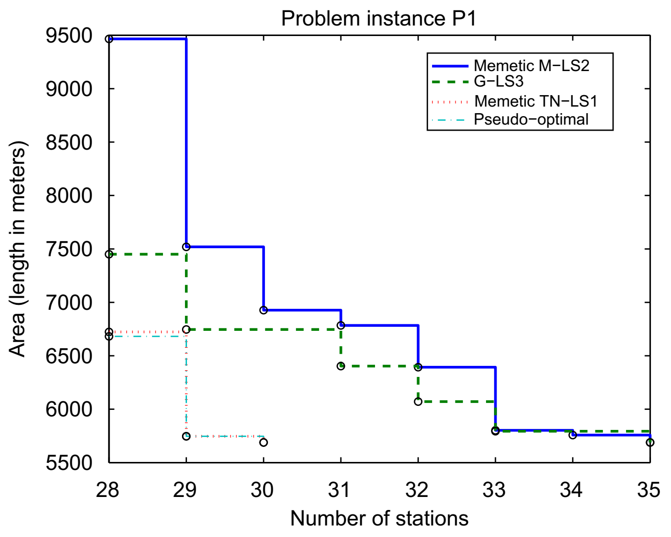


Fig. 10. Boxplots represent the following I_ε dominance probabilities for P1 to P9: (M-G) $P_{\text{MACS-LS2}}$(GRASP-LS3), (G-M) $P_{\text{GRASP-LS3}}$(MACS-LS2), (M-N) $P_{\text{MACS-LS2}}$(NSGA-II-LS1), (N-M) $P_{\text{NSGA-II-LS1}}$(MACS-LS2), (G-N) $P_{\text{GRASP-LS3}}$(NSGA-II-LS1), and (N-G) $P_{\text{NSGA-II-LS1}}$(GRASP-LS3).

Nissan Pathfinder engine (shown in Fig. 12) at the plant of Barcelona (Spain). The assembly of these engines is divided into 378 operation tasks, although we have grouped these operations into 140 different tasks. The available cycle time is 180 s. More information can be found at http://www.nissanchair.com/TSALBP.

Appendix A reports the details about the tackled Nissan instance, which is originated in the final assembly phase of the Nissan Pathfinder engines. It shows the task number (n), internal identifier from NISSAN (id.), duration of the task (t) in seconds, required area (a) in metres, and precedence constraints of each task. Some changes have been made to the original data which are described as follows:

- The original line corresponds to a mixed-model assembly line. Following the procedure in use in Nissan, the duration of tasks has been modified taking into account the expected production mix of the variants to assemble. Notice that, the production mix does not alter the area required for each task.
- The space requirements originated by tools and machinery required for the assembly have been omitted. Due to the similitude of the tasks and the low cost of the used machinery, each workstation is considered to contain all the tools. Thus,



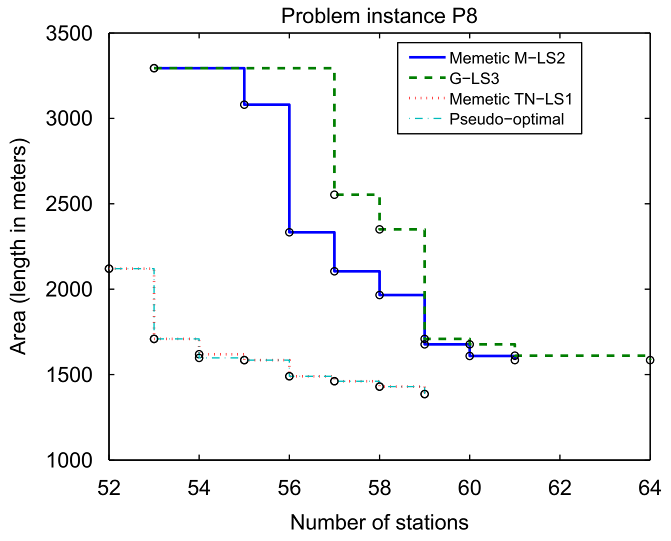


Fig. 11. Attainment surface plots of the best variant of each of the three memetic designs for instances P1 and P8.

Table 3
p-values and statistical significance (represented by a symbol "+", "−", or "=") of the best three MA approaches for the 9 problem instances. TN is the *advanced TSALBP-NSGA-II*.

	MACS-LS2	GRASP-LS3	TN-LS1
MACS-LS2	•	0.1659 =	0.000051 −
GRASP-LS3	0.1659 =	•	0.000057 −
TN-LS1	0.000051 +	0.000057 +	•

the space required for them can be subtracted from the total available space for a workstation.

- The duration, required area, and precedence constraints of tasks have been slightly altered due to confidentiality issues.

4.3.2. Analysis of the results of the proposed memetic approaches

As done with the real-like instances, we have analysed the performance of the different memetic designs and variants proposed. We have compared six memetic MACS variants, three GRASP methods, and six memetic variants of the *advanced TSALBP-NSGA-II*. The *HVR* values of the algorithms can be seen in Table 4 and the I_ε values of the boxplots in Fig. 13. In the next paragraphs the results obtained by the algorithms are analysed.

Fig. 12. The Nissan PathFinder engine. It consists of 747 pieces and 330 parts.

Table 4
Mean and standard deviation $\overline{x}(\sigma)$ of the *HVR* performance indicator values for the best variants of MACS, GRASP, and *advanced TSALBP-NSGA-II* MAs in the Nissan case study. Higher values indicate better performance. Underlined values are the best results of each algorithm while bold values correspond to the global best results.

Memetic MACS algorithm		GRASP		Memetic advanced TSALBP-NSGA-II	
M	0.7993 (0.007)	**G**	0.7562 (0.01)	**TN**	0.7043 (0.056)
M-LS1	0.9413 (0.007)	**G-LS1**	<u>0.8999 (0.005)</u>	**TN-LS1**	0.9717 (0.006)
M-LS2	<u>0.9428 (0.006)</u>	**G-LS2**	<u>0.8999 (0.005)</u>	**TN-LS2**	<u>**0.9773 (0.006)**</u>
M-LS3	<u>0.9428 (0.006)</u>	**G-LS3**	0.8993 (0.006)	**TN-LS3**	<u>**0.9773 (0.006)**</u>
M-LS4	0.9124 (0.007)			**TN-LS4**	0.9071 (0.038)
M-LS5	0.9108 (0.008)			**TN-LS5**	0.9083 (0.038)
M-LS6	0.9108 (0.008)			**TN-LS6**	0.9083 (0.038)

Memetic MACS algorithm: We can reach the following conclusions:

- The memetic variants of the MACS algorithm improve the performance of the MACS algorithm. The difference is clear both in the *HVR* values and in the boxplots.
- As happened with the preliminary experimentation, the memetic MACS variants that applied the LS methods to all the solutions (M-LS1, M-LS2, M-LS3) are better than the remainder (M-LS4, M-LS5, M-LS6).
- Among the first three memetic MACS variants, M-LS2 and M-LS3 are those achieving the best results according to the used performance indicators. Therefore, an intermediate value between 20 iterations (M-LS1) and 100 iterations (M-LS3) is enough to lead to a proper convergence.

GRASP: Again, variants including LS clearly outperform the basic algorithm (first stage of the GRASP in this case). The best convergence is obtained by G-LS1 and G-LS2 with a low difference with respect to the third option. Then, there is not a need for a high number of iterations to provide the best results, 20 iterations are enough. In fact, the highest exploitation value (100 iterations) slightly decreases the performance of the algorithm.

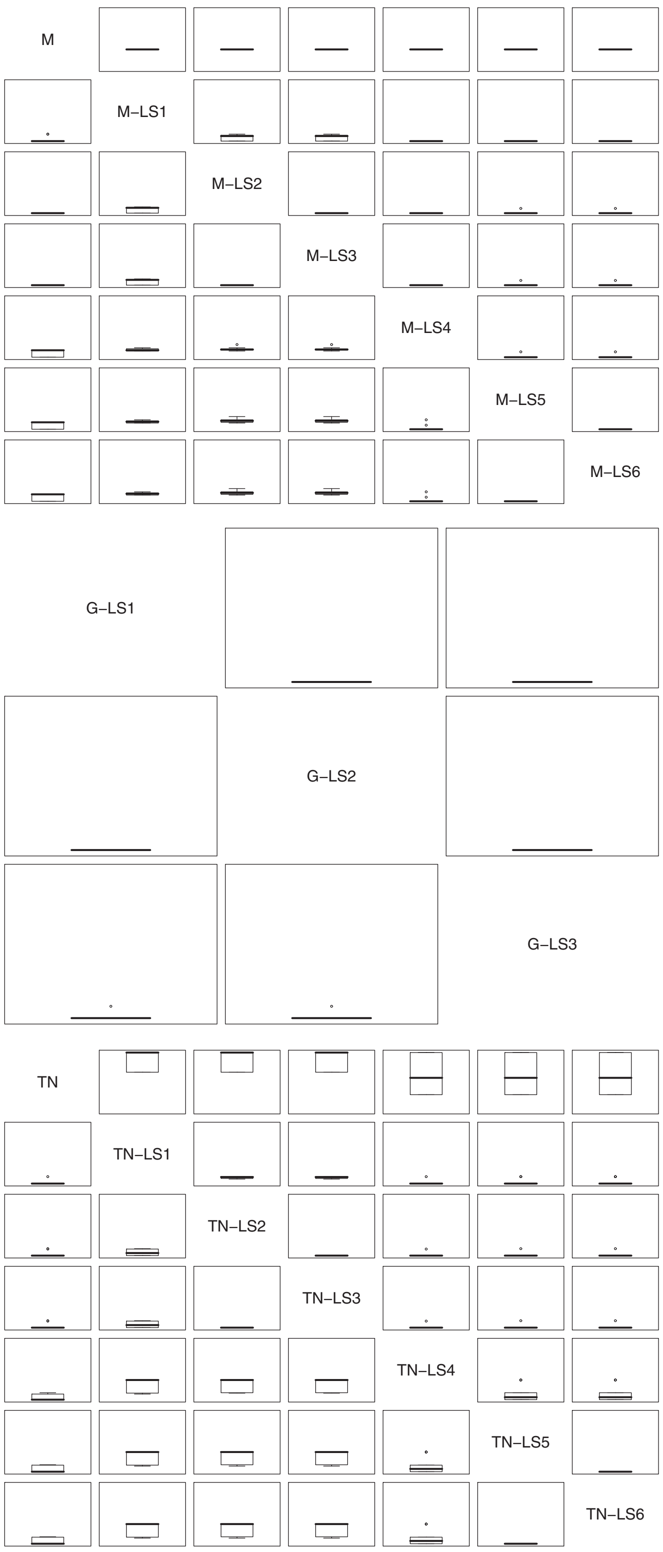
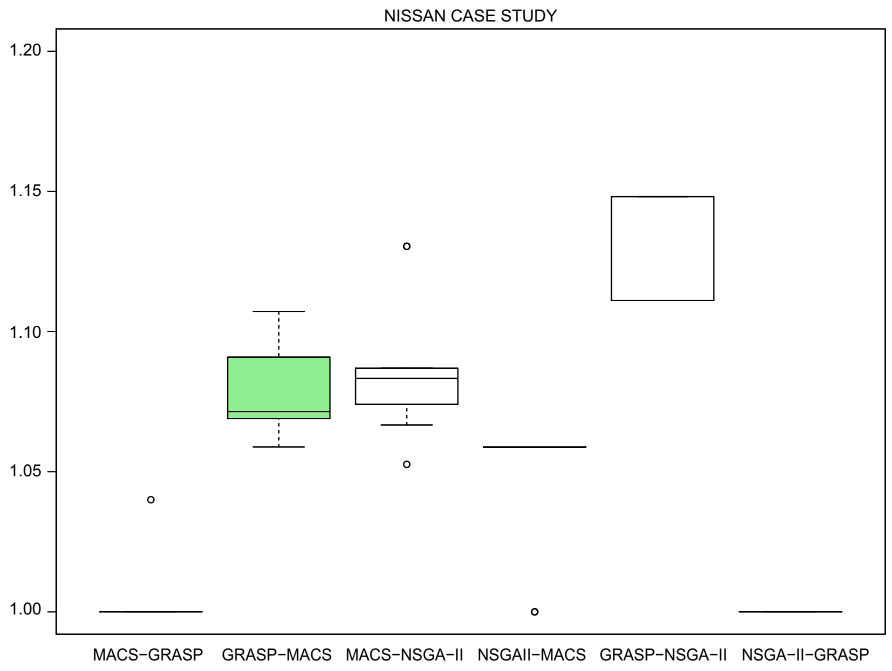


Fig. 13. I_{ε} values represented by means of boxplots comparing the memetic variant of each of the three memetic designs for the Nissan case study.

Memetic advanced TSALBP-NSGA-II : The following items summarise the obtained conclusions:

- The performance of the memetic variants is again much better than the TSALBP-NSGA-II in all the performance indicators.
- As happened with the memetic MACS algorithms, the variants that apply the LS to all the solutions outperform those based on the selective LS application. Consequently, TN-LS1, TN-LS2, and TN-LS3 are also better than TN-LS4, TN-LS5, and TN-LS6 in the Nissan case study.
- However, for the *advanced TSALBP-NSGA-II*, more than 20 iterations are needed to achieve the best performance as T-LS2 and T-LS3 results improve T-LS1 ones. This situation is equivalent to the memetic MACS algorithms in the Nissan case study but differs from what happened for the same MA designs in the experiments developed in Section 4.2.



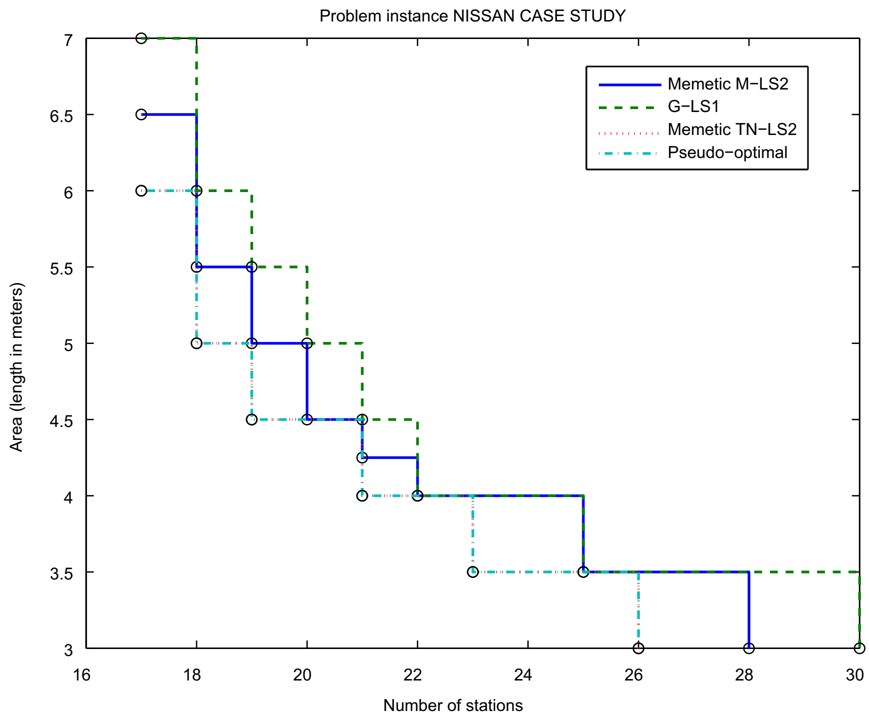


Fig. 14. I_ε boxplots and attainment surface plot of the best variants of the MAs for the Nissan case study.

4.3.3. Global analysis and final benchmarking

The global conclusions for the Nissan case study are basically that: (a) the memetic variants in all the algorithms are better than the basic global search with a significant performance difference (i.e. proper memetic designs have been achieved); (b) the MAs that apply the LS to all the solutions are always better than the rest; (c) there is not a great difference between the number of iterations used in the algorithms, but normally a trade-off value of 50 iterations is the most appropriate.

For the final benchmarking we have selected the best MA for each global search method as done in the preliminary study. Table 4 shows the *HVR* values of the memetic MACS-LS2 algorithm, the GRASP-LS1, and the memetic TN-LS2. The boxplots of the I_ε performance indicator comparing the latter three algorithms as well as their attainment surface plots are represented in Fig. 14.

Dominance probabilities based on the I_ε performance indicator comparisons between the best algorithms are calculated and represented using the boxplots in Fig. 15. Wilcoxon statistical test is also applied for the Nissan case study as done with the real-like instances in Section 4.2.3. The results of the test are shown in Table 5.

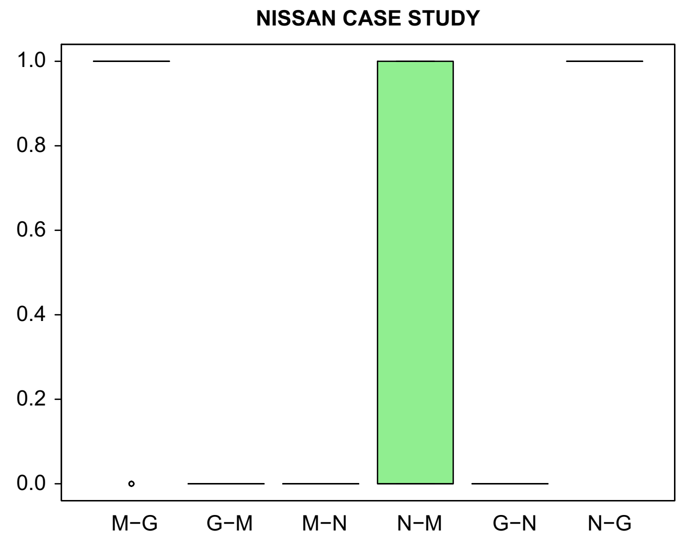


Fig. 15. Boxplots represent the following I_ε dominance probabilities for the Nissan case study: (M-G) $P_{\text{MACS-LS2}}$(GRASP-LS1), (G-M) $P_{\text{GRASP-LS1}}$(MACS-LS2), (M-N) $P_{\text{MACS-LS2}}$(NSGA-II-LS2), (N-M) $P_{\text{NSGA-II-LS2}}$(MACS-LS2), (G-N) $P_{\text{GRASP-LS1}}$(NSGA-II-LS2), and (N-G) $P_{\text{NSGA-II-LS2}}$(GRASP-LS1).

Table 5
p-values and statistical significance (represented by a symbol "+", "−", or "=") of the best three MA approaches for the Nissan case study. TN is the *advanced TSALBP-NSGA-II*.

	MACS-LS2	**GRASP-LS1**	**TN-LS2**
MACS-LS2	•	0.0004 +	0.0767 =
GRASP-LS1	0.0004 −	•	0.000016 −
TN-LS2	0.0767 =	0.000016 +	•

In view of the results provided by all these indicators we can conclude that the memetic *advanced TSALBP-NSGA-II* is the best algorithm to deal with the real-world Nissan instance, reaching almost all the solutions in the pseudo-optimal Pareto front (see Fig. 14), and obtaining better I_ε values, dominance probabilities (Fig. 15), and *HVR* values than the remainder. The *advanced TSALBP-NSGA-II* is also significantly better than GRASP-LS1 according to the statistical test shown in Table 5. Although there is no statistical significance with respect to MACS-LS2 (see symbol "=" in the corresponding cells of the table), the obtained *p*-value is very close to the considered significance level (0.05). In addition, Fig. 15 clearly shows how the *advanced TSALBP-NSGA-II* is outperforming MACS-LS2 on several comparisons while the latter is never able to do so.

The memetic MACS algorithm is the second algorithm in performance. It converges better than the GRASP-LS1 and its difference is statistically significant (see the statistical test results in Table 5). GRASP-LS1 is finally the worst performing algorithm.

5. Concluding remarks and future works

In this contribution, we have successfully proposed novel memetic designs to solve the TSALBP-1/3. The new MAs to tackle this industrial problem are multiobjective and make use of a multi-criteria LS procedure with two problem-specific neighbourhood operators, one per objective. The proposals are based on three different global search methods: a MACS algorithm, a GRASP, and an advanced NSGA-II-based technique for the TSALBP-1/3.

We have studied different variants to analyse the impact of the intensification and diversification induced by the multi-criteria LS on the performance of the MAs when solving nine realistic and one real-life problem instance. From this study, we have concluded that the LS is more powerful if it is applied to all the generated solutions and not just to a reduced number of them (a 0.0625 percent of the solutions). In addition, the LS depth, i.e. the number of iterations to be considered for the LS,

Table 6
Problem instance from Nissan Pathfinder motor engine assembly line balancing. Number (*n*), internal identifier (id.), operation time (*t*), required area (*a*) and set of immediately predecessor tasks (P) are given for each task.

n	Id.	*t*	*a*	P	*n*	id.	*t*	*a*	P
1	50100	60	3		2	50110	75	2	3,31
3	50120	20	0.5	1	4	50500	60	1	3,5
5	50501	20	0.5	1	6	50600	60	1.5	4,5
7	50800	45	1	1	8	50900	10	0.5	1
9	51000	20	0.5	1	10	51200	30	0.5	1
11	51400	15	0.5	1	12	51401	15	0.5	11
13	51600	15	1	1	14	51800	10	0.5	3,13
15	52000	8	1	9,10,11,13,14	16	52010	8	0.5	9,10,11,13,14
17	52200	80	1	9,10,11,13,14	18	52400	40	0.5	9,10,11,13,14
19	52600	5	0.5	9,10,11,13,14	20	52610	5	0.5	9,10,11,13,14
21	52650	5	0.5	9,10,11,13,14	22	52700	7	0.5	26,27
23	52710	7	0.5	26,27	24	52720	30	0.5	26,27
25	52730	30	0.5	26,27	26	52750	5	0.5	15,16,17,18,19,20,21
27	52760	5	0.5	15,16,17,18,19,20,21	28	52800	30	1	22,23,24,25
29	52820	10	0.5	28	30	52900	15	1	29
31	52901	10	0	6,7,8,30	32	53050	15	0.5	31
33	53100	30	1	32	34	53200	10	0.5	32
35	53300	5	0.5	36	36	53301	25	1	32
37	53400	15	0	32,35	38	53600	5	0.5	33,34,36,37
39	53630	5	0.5	33,34,36,37	40	53650	5	0.5	33,34,36,37
41	54000	60	0.5	38,39,40	42	54100	15	1.5	38,39,40
43	54120	15	1.5	38,39,40	44	54200	25	0.5	41,42,43
45	54210	25	0.5	41,42,43	46	54230	5	0.5	44,45
47	54240	35	0.5	46	48	54250	35	0.5	46
49	54260	5	0.5	42,43	50	54270	15	0.5	47,48,49
51	54280	25	0	47,48,49	52	54290	30	0	47,48,49
53	54300	15	0	47,48,49	54	54310	15	0	47,48,49
55	54320	20	0	47,48,49	56	54330	10	0	47,48,49

Table 6 (*continued*)

n	Id.	*t*	*a*	P	*n*	id.	*t*	*a*	P
57	54370	10	0.5	50,51,52,53,54,55,56	58	54500	20	0.5	57,59,60
59	54501	5	0	41	60	54520	20	0.5	42,43
61	54700	45	1	57,58	62	54720	30	0.5	61
63	54800	30	0.5	57	64	54820	10	0.5	57
65	55050	5	0	61,62,63,64	66	55200	10	0.5	61,62,63,64
67	55250	15	0.5	66	68	55300	60	1.5	65,67
69	55350	10	0.5	68	70	55400	30	1	67
71	55500	10	0.5	68	72	55540	10	0.5	68
73	55800	40	1.5	71,72	74	55900	25	0.5	68,69,70,73
75	56000	10	0.5	74	76	56020	10	1	74
77	56100	15	0.5	75	78	56200	15	0.5	79
79	56220	15	0.5	74	80	56300	10	0.5	76,77,78
81	56400	10	1	76,77,78	82	56401	10	0	80,81
83	56420	20	0.5	82	84	56430	10	0	83
85	56440	20	0.5	75,84	86	56500	25	0.5	82
87	56600	20	0.5	82	88	56700	15	0.25	84
89	56750	20	0.5	88	90	56760	30	0.5	88
91	56800	20	0.5	85,86,87,88	92	56880	25	0.5	89,90,91
93	56900	10	0.5	92	94	56920	5	0.5	89,90,91
95	56940	20	0.5	94	96	57000	10	0.5	93,95,99
97	57050	5	0.5	93,95,99	98	57100	80	0	92
99	57120	30	0	89,90,91	100	57150	10	0.5	98,99
101	57160	10	0.5	98,99	102	57200	20	0.5	100,101
103	57210	30	0.5	100,101	104	57250	5	0	102,103
105	57300	30	0.5	106	106	57301	25	0.5	100,101
107	57400	5	0	100,101,104	108	57450	5	0	100,101,104
109	57500	5	0.5	108	110	57505	5	0	108
111	57510	10	0	109,110	112	57520	10	0	109,110
113	57530	15	0.5	108	114	57540	20	0	113
115	57550	20	0	113	116	57700	45	1	111,112,114,115
117	57900	20	0.5	118	118	57950	25	0	116
119	58000	25	0	116	120	58050	20	0.5	119
121	58200	45	1.5	105,107,117,120	122	58201	15	0.5	121
123	58250	10	0.5	122	124	58300	10	0	123
125	58310	20	1	124	126	58350	30	0.5	125
127	58351	10	0.5	126	128	58400	25	0.5	117,120
129	58500	30	0.5	126	130	58900	30	0.75	127,128,129
131	59000	40	0.5	117,120	132	59100	25	1	131
133	59300	25	0.5	130	134	59320	20	0.5	132
135	59340	15	0.5	134	136	59400	20	0.5	135
137	59500	30	0.5	136	138	59510	30	0.5	136
139	59600	15	1	137,138	140	59900	120	0	133,139

was also studied. The behaviour of this parameter depends on the problem instance and the search capabilities of the global search method. When the latter method already shows a good intensification–diversification trade-off, the resulting MA will perform better with a low number of LS iterations. We can state that, in this experimentation, there is not a need to perform more than 50 iterations in any case.

Apart from the LS study, the three memetic designs were compared to each other. The memetic *advanced TSALBP-NSGA-II* showed its excellent performance, obtaining the best solutions. The second MA in quality was not clear enough since the memetic MACS and GRASP performed differently depending on the problem instance. The memetic *advanced TSALBP-NSGA-II* was again the best approach to deal with the real instance of the Nissan industry plant in Barcelona, obtaining outstanding results.

Future work is devoted to: (i) apply preferences in the algorithms by means of interactive procedures, (ii) deal with the combined three-objective optimisation of cycle time, area, and number of stations, and (iii) study the use of other MOACO algorithms to solve the problem.

Acknowledgements

This work has been supported by the UPC Nissan Chair and the Spanish Ministerio de Educación y Ciencia under the PROTHIUS-III project (DPI2010-16759) and by the Spanish Ministerio de Ciencia e Innovación under project TIN2009-07727, both including EDRF fundings.

Appendix A. Description of the Nissan Pathfinder instance

The assembly line of the Nissan Pathfinder is distributed serially where nine types of engines ($p_1,\ldots,p_9$) with different characteristics are assembled. The first three engines are for 4×4 vehicles, the last four for trucks of medium weight, and the models p_4 and p_5 are user for vans.

Further information about the tasks of the assembly line is reported in Table 6.

References

Barán, B., Schaerer, M., 2003. A multiobjective ant colony system for vehicle routing problem with time windows. In: 21st IASTED International Conference, Innsbruck, Germany, pp. 97–102.

Bautista, J., Pereira, J., 2007. Ant algorithms for a time and space constrained assembly line balancing problem. European Journal of Operational Research 177, 2016–2032.

Baybars, I., 1986. A survey of exact algorithms for the simple assembly line balancing problem. Management Science 32 (8), 909–932.

Boysen, N., Fliedner, M., Scholl, A., 2008. Assembly line balancing: which model to use when? International Journal of Production Economics 111, 509–528.

Chankong, V., Haimes, Y.Y., 1983. Multiobjective Decision Making Theory and Methodology. North-Holland.

Chica, M., Cordón, O., Damas, S., Bautista, J., 2010a. Multiobjective, constructive heuristics for the 1/3 variant of the time and space assembly line balancing problem: ACO and random greedy search. Information Sciences 180, 3465–3487.

Chica, M., Cordón, O., Damas, S., Bautista, J., 2010b. A multiobjective GRASP for the 1/3 variant of the time and space assembly line balancing problem. In: Trends in Applied Intelligent Systems, Lecture Notes in Artificial Intelligence, vol. 6098. pp. 656–665.

Chica, M., Cordón, O., Damas, S., 2011a. An advanced multi-objective genetic algorithm design for the time and space assembly line balancing problem. Computers and Industrial Engineering 61 (1), 103–117. doi:10.1016/j.cie.2011.03.001.

Chica, M., Cordón, O., Damas, S., Bautista, J., 2011b. Including different kinds of preferences in a multi-objective ant algorithm for time and space assembly line balancing on different nissan scenarios. Expert Systems with Applications 38, 709–720.

Chica, M., Cordón, O., Damas, S., Bautista, J., Pereira, J., 2008. Incorporating preferences to a multi-objective ant algorithm for time and space assembly line balancing. Ant Colony Optimization and Swarm Intelligence, Lecture Notes in Computer Science, vol. 5217. Springer, Berlin, Germany, pp. 331–338.

Coello, C.A., Lamont, G.B., Van Veldhuizen, D.A., 2007. Evolutionary Algorithms for Solving Multi-objective Problems, second ed. Springer.

Deb, K., 2001. Multi-objective Optimization using Evolutionary Algorithms. Wiley.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6 (2), 182–197.

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

Dorigo, M., Stützle, T., 2004. Ant Colony Optimization. MIT Press, Cambridge.

Feo, T.A., Resende, M.G.C., 1995. Greedy randomized adaptive search procedures. Journal of Global Optimization 6, 109–133.

Fonseca, C.M., Fleming, P.J., 1996. On the performance assessment and comparison of stochastic multiobjective optimizers. In: Proceedings of the Fourth International Conference on Parallel Problem Solving from Nature (PPSN), Lecture Notes in Computer Science, vol. 1141. Berlin, Germany, pp. 584–593.

Gandibleux, X., Freville, A., 2000. Tabu search based procedure for solving the 0–1 multiobjective knapsack problem: the two objectives case. Journal of Heuristics 6 (3), 361–383.

García, S., Molina, D., Lozano, M., Herrera, F., 2009. 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, 617–644.

Hansen, M.P., 1997. Tabu search for multiobjective optimization: MOTS. In: 13th International Conference on Multiple Criteria Decision Making, MCDM'97, Cape Town, South Africa.

Hart, W.E., 1994. Adaptive global optimization with local search. Ph.D. Thesis, University of Califorina, San Diego.

Herrera, F., Lozano, M., Molina, D., 2005. Continuous scatter search: an analysis of the integration of some combination methods and improvement strategies. European Journal of Operational Research 169 (2), 450–476.

Ishibuchi, H., Narukawa, K., Tsukamoto, N., Nojima, Y., 2008. An empirical study on similarity-based mating for evolutionary multiobjective combinatorial optimization. European Journal of Operational Research 188 (1), 57–75.

Ishibuchi, H., Yoshida, T., Murata, T., 2003. Balance between genetic search and local search in memetic algorithms for multiobjective permutation flow shop scheduling. IEEE Transactions on Evolutionary Computation 7 (2), 204–223.

Jaszkiewicz, A., 2002. Genetic local search for multiple objective combinatorial optimization. European Journal of Operational Research 137 (1), 50–71.

Knowles, J., 2006. ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Transactions on Evolutionary Computation 10 (1), 50–66.

Knowles, J., Corne, D., 2002. On metrics for comparing nondominated sets. In: Proceedings of the 2002 Congress on Evolutionary Computation (CEC), vol. 1, Honolulu, HI, USA, pp. 711–716.

Knowles, J., Corne, D., 2003. Instance generators and test suites for the multiobjective quadratic assignment problem. Proceedings of Evolutionary Multicriterion Optimization (EMO 2003), Lecture Notes in Computer Science, vol. 2632. Springer-Verlag, Berlin, Germany, pp. 295–310.

Krasnogor, N., Smith, J., 2000. A memetic algorithm with self-adaptive local search: TSP as a case of study. In: Genetic and Evolutionary Computation Conference, GECCO'05, pp. 987–994.

Lozano, M., Herrera, F., Krasnogor, N., Molina, D., 2004. Real-coded memetic algorithm with crossover hill-climbing. Evolutionary Computation 12, 273–302.

Moscato, P., 1989. On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms. Technical Report 826, Caltech Concurrent Computation Program, Pasadena.

Noman, N., Iba, H., 2005. Enhancing differential evolution performance with local search for high dimensional function optimization. In: Genetic and Evolutionary Computation Conference, GECCO'05, pp. 967–974.

Ong, Y.S., Lim, M., Zhu, N., Wong, K., 2006. Classification of adaptive memetic algorithms: a comparative study. IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics 36 (1), 141–152.

Ong, Y.S., Lim, M., Zhu, N., Wong, K., 2010. Memetic computation—past, present and future. IEEE Computational Intelligence Magazine 5 (2), 24–31.

Paquete, L., Stützle, T., 2006. A study of stochastic local search algorithms for the biobjective QAP with correlated flow matrices. European Journal of Operational Research 169, 943–959.

Pérez-Bellido, A.M., Salcedo-Sanz, S., Ortiz-García, E.G., Portilla-Figueras, J.A., López-Ferreras, F., 2008. A comparison of memetic algorithms for the spread spectrum radar polyphase codes design problems. Engineering Applications of Artificial Intelligence 21 (8), 1233–1238.

Pishvaee, M.S., Farahani, R.Z., Dullaert, W., 2010. A memetic algorithm for bi-objective integrated forward/reverse logistics network design. Computers and Operations Research 37 (6), 1100–1112.

Prins, C., 2009. Two memetic algorithms for heterogeneous fleet vehicle routing problems. Engineering Applications of Artificial Intelligence 22, 916–928.

Rachamadugu, R., Talbot, B., 1991. Improving the equality of workload assignments in assembly lines. International Journal of Production Research 29, 619–633.

Sabuncuoglu, I., Erel, E., Tayner, M., 2000. Assembly line balancing using genetic algorithms. Journal of Intelligent Manufacturing 11, 295–310.

Sánchez, L., Villar, J.R., 2008. Obtaining transparent models of chaotic systems with multi-objective simulated annealing algorithms. Information Sciences 178, 950–970.

Santamaría, J., Cordón, O., Damas, S., García-Torres, J.M., Quirin, A., 2009. Performance evaluation of memetic approaches in 3D reconstruction of forensic objects. Soft Computing 13 (8–9), 883–904.

Scholl, A., 1999. Balancing and Sequencing of Assembly Lines, second ed. Physica-Verlag, Heidelberg.

Scholl, A., Becker, C., 2006. State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. European Journal of Operational Research 168 (3), 666–693.

Scholl, A., Voss, S., 1996. Simple assembly line balancing—heuristic approaches. Journal of Heuristics 2, 217–244.

Teghem, J., Jaszkiewicz, A., 2003. Multiple objective metaheuristics for combinatorial optimization: a tutorial. In: Proceedings of the Fourth Metaheuristic International Conference, MIC 2003, Kyoto, Japan, pp. 25–28.

Wolpert, D.H., Macready, W.G., 1997. No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1 (1), 67–82.

Zitzler, E., Deb, K., Thiele, L., 2000. Comparison of multiobjective evolutionary algorithms: empirical results. Evolutionary Computation 8 (2), 173–195.

Zitzler, E., Thiele, L., 1999. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Transactions on Evolutionary Computation 3 (4), 257–271.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, V.G., 2003. Performance assessment of multiobjective optimizers: an analysis and review. IEEE Transactions on Evolutionary Computation 7 (2), 117–132.