A Multiobjective Memetic Ant Colony Optimization Algorithm for the 1/3 Variant of the Time and Space Assembly Line Balancing Problem

Manuel Chica*, Oscar Cordon*, Sergio Damas* and Joaquín Bautista†
*European Centre for Soft Computing
c/ Gonzalo Gutiérrez Quirós s/n, Mieres (Asturias), Spain
Email: {manuel.chica, oscar.cordon, sergio.damas}@softcomputing.es
†Universitat Politècnica de Catalunya - Nissan Chair (http://www.nissanchair.com)
Barcelona, Spain. Email: joaquin.bautista@upc.edu

***Abstract*—Time and space assembly line balancing considers realistic multiobjective versions of the classical assembly line balancing industrial problems, involving the joint optimization of conflicting criteria such as the cycle time, the number of stations, and/or the area of these stations. The aim of this contribution is to present a new multiobjective memetic algorithm based on ant colony optimization for the 1/3 variant of this family of industrial problems. This variant involves the joint minimisation of the number and the area of the stations, given a fixed cycle time limit. The good behaviour of the proposal is shown in nine problem instances.**

I. Introduction

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 [1] [2]. Taking this family as a base and adding spatial information to enrich it, Bautista and Pereira recently proposed a more realistic framework: the time and space assembly line balancing problem (TSALBP) [3]. This framework considers an additional space constraint to become a simplified version of real-world problems. The new space constraint emerged due to the study of the specific characteristics of the Nissan plant in Barcelona (Spain).

As many real-world problems, TSALBP formulations have a multicriteria nature [4] 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 minimise the number of stations and their area for a given product cycle time. TSALBP-1/3 has an important set of hard constraints like precedences or cycle time limits for each station. Thus, the use of constructive approaches is more convenient than others like local or global search procedures [5]. In [6] we successfully tackled the TSALBP-1/3 by means of a specific procedure based on the Multiple Ant Colony System (MACS) algorithm [7], that approach is the state-of-the-art of TSALBP-1/3. Later in [8], a multiobjective GRASP method [9] was presented also showing the appropriateness of using local search (LS) operators with multiobjective metaheuristics to solve the problem.

The term memetic algorithm (MA) was introduced by Moscato to describe genetic algorithms where LS played a significant role [10]. This "hybrid" metaheuristic has demonstrated its good performance because of the combination of the genetic operators, that present a global search behaviour, and the local optimizer, which acts to improve the solutions produced by the genetic operators. With this methodology, the LS strategy is part of the whole evolutionary procedure. From the original contribution of Moscato, the evolutionary computation community has shown a great interest on MAs resulting in a broad research area [11], [12]. More specifically, MAs have been widely used in industrial and engineering applications [13], [14].

However, the use of multiobjective local search operators to improve the solutions obtained by a global search procedure for the assembly line balancing has not been extensively explored. In this paper we do it by extending the multiobjective ant colony optimization algorithm, MACS [6], by means of incorporating a multicriteria local search scheme. An experimentation is carried out in nine problem instances, comparing the behaviour of the memetic MACS algorithm with the basic MACS proposal and the multiobjective GRASP method. Performance indicators are used to analyse the behaviour of the algorithms

The paper is structured as follows. In Section II, the problem formulation is explained. Then, the MACS metaheuristic is described in Section III, and the new multicriteria local search structure is shown in Section IV. The experimentation setup as well as the analysis of results are presented in Section V. Finally, some concluding remarks and future research are discussed in Section VI.

978-1-61284-333-9/11/$26.00 ©2011 IEEE

II. The Time and Space Assembly Line Balancing Problem

The manufacturing of a production item is divided into a set V of n tasks. Each task j requires an operation time for its execution $t_j > 0$ that is determined as a function of the manufacturing technologies and the employed resources. Each station k is assigned to a subset of tasks S_k ($S_k \subseteq V$), called workload. A task j is assigned to a station k.

Each task j has a set of direct predecessors, P_j, which must be accomplished before starting it. These constraints are normally represented by means of an acyclic precedence graph, whose vertices stand for the tasks and where a directed arc (i, j) indicates that task i must be finished before starting task j on the production line. Thus, if $i \in S_h$ and $j \in S_k$, then $h \leq k$ must be fulfilled. Each station k presents a station workload time $t(S_k)$ that is equal to the sum of the tasks' lengths assigned to the station k. SALBP [2] focuses on grouping tasks in workstations by an efficient and coherent way. There is a large variety of exact and heuristic problem-solving procedures for it [15].

The need of introducing space constraints in the assembly lines' design is based on two main reasons: (a) the length of the workstation is limited in the majority of the situations, and (b) the required tools and components to be assembled should be distributed along the sides of the line. Hence, an area constraint may be considered by associating a required area a_j to each task j and an available area A_k to each station k that, for the sake of simplicity, we shall assume it to be identical for every station and equal to $A : A = max_{\forall k \in \{1..n\}}\{A_k\}$. Thus, each station k requires a station area $a(S_k)$ that is equal to the sum of areas required by the tasks assigned to station k.

This leads us to a new family of problems called TSALBP in [3]. It may be stated as: given a set of n tasks with their temporal t_j and spatial a_j attributes ($1 \leq j \leq n$) and a precedence graph, each task must be assigned to a single station such that: (i) every precedence constraint is satisfied, (ii) no station workload time ($t(S_k)$) is greater than the cycle time (c), and (iii) no area required by any station ($a(S_k)$) is greater than the available area per station (A).

TSALBP presents eight variants depending on three optimization criteria: m (the number of stations), c (the cycle time) and A (the area of the stations). Within these variants there are four multiobjective problems and we will tackle one of them, 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, mathematically formulated as follows:

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

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

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

We chose this variant because it is realistic in the automotive industry since the annual production of an industrial plant (and therefore, the cycle time c) is usually set by some market objectives. For more information we refer the interested reader to [6].

III. MACS

MACS [7] was proposed as an extension of ant colony system (ACS) [16] to deal with multiobjective problems. In [6], 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 [6] 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 [2] to face the precedence problem (as usually done for the SALBP [15]). 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 $(station_k, task_j)$, $k = 1...n$, $j = 1...n$, with n being the number of tasks, so the pheromone trail matrix has a bi-dimensional nature. Since MACS is Pareto-based, the pheromone trails are updated using the current non-dominated solution set (Pareto 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:

$$p\,(closing\ k) = \frac{\sum_{i \in S_k} t_i}{c}. \tag{3}$$

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 opened. 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:

$$j = \begin{cases} \arg\ \max_{j\in\Omega}(\tau_{ij} \cdot [\eta_{ij}^0]^{\lambda\beta} \cdot [\eta_{ij}^1]^{(1-\lambda)\beta}), & \text{if } q \leq q_0, \\ \hat{i}, & \text{otherwise.} \end{cases} \quad (4)$$

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

$$p(j) = \begin{cases} \frac{\tau_{ij}\cdot[\eta_{ij}^0]^{\lambda\beta}\cdot[\eta_{ij}^1]^{(1-\lambda)\beta}}{\sum_{u\in\Omega}\tau_{iu}\cdot[\eta_{iu}^0]^{\lambda\beta}\cdot[\eta_{iu}^1]^{(1-\lambda)\beta}}, & \text{if } j \in \Omega, \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

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 $< i, j >$ using the average costs of the τ_0 value. It is done as follows:

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \rho \cdot \tau_0 \quad (6)$$

The interested reader is referred to [6] for a complete description of the MACS proposal for the TSALBP-1/3.

IV. Multicriteria LS Structure and Components

Mainly, there are two stochastic LS approaches for multi-objective combinatorial optimization problems [17], [18]. 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) [19], [20]. The second family is based on considering different scalarizations of the objective function vector [21], [22], [23]. The MA design 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:

$$Min\ (\lambda^1 A + \lambda^2 m). \quad (7)$$

This will be the function to be optimised by the multi-criteria LS of the memetic MACS. As usually done in the multiobjective MA area (see for example [23]), the weight vector $\lambda = (\lambda^1, \lambda^2)$ is created at random for each constructed solution.

The existing local improvement procedures for ALB are based on moves [24]. The LS operators are based on such moves of tasks. In our design, two different neighbour generation operators will be considered and selected depending on the weight vector λ (see Section IV). If $\lambda^1 > \lambda^2$, the neighbour operator for minimising the A objective will be followed since the LS optimization 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, the first, ES_j, and last, LP_j, station where task j may be assigned according to the 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 $k2 \in [ES_j, LP_j]$.

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.

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 branch & bound function (Algorithm 3) to search for a feasible solution having the Target_Station's tasks reallocated in other stations.

The local search is also run $MAX\_ITERATIONS = 20$ times. In addition, we have to specify a maximum number of stations ($MAX\_STATIONS$) to limit the computational time of the local search.

V. Experiments

We explain the instances, parameters and performance indicators used for the experimentation. Then, we analyse the results of the different algorithms.

A. Problem Instances and Parameter Values

Nine problem instances with different features have been selected for this first experimentation: `arc111` with cycle

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

```
while Iterations ≤ MAX_ITERATIONS do
    Target_Station ← Find the station with the highest area;
    Tasks ← Descending_Sort(tasks of Target_Station);
    while no scalarization function improvement AND Tasks ≠ ∅ do
        Task ← First element of Set_Of_Tasks ;
        Find First_Station and Last_Station of Task;
        while no scalarization function improvement do
            Possible_Station ← station with the lowest area ∈ [First_Station,Last_Station];
            Move Task from Target_Station to Possible_Station;
            if scalarization function improvement then
                Make the move permanent;
            end
        end
        Remove Task from Set_Of_Tasks;
    end
    if Target_Station = ∅ then
        Remove Target_Station;
    end
    Iterations ← Iterations +1;
end
return true if scalarization function is improved;
```

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

```
while Iterations ≤ MAX_ITERATIONS do
    Set_Of_Stations ← Ascending Sort (with respect to no. of tasks);
    i ← 1;
    while i ≤ MAX_STATIONS AND no scalarization function improvement do
        Target_Station ← i-th element of Set_Of_Stations;
        Set_Of_Tasks ← Descending_Sort(tasks of Target_Station);
        for all elements of Set_Of_Tasks do
            Find First_Station and Last_Station;
        end
        First_Element = First Element of Set_Of_Tasks
        BB(First_Element, Set_Of_Tasks);
        if no scalarization function improvement then
            i ← i + 1;
        end
    end
    Iterations ← Iterations +1;
end
return true if scalarization function is improved;
```

Algorithm 3: The pseudo-code of the Branch & Bound function used by the LS operator for objective m.

```
Function BB(Current_Task, Set_Of_Tasks)
if all elements of Set_Of_Tasks allocated then
    // Base case
    Calculate scalarization function of the objective function vector;
else
    for all the possible stations of Current_Task do
        Move Current_Task to the selected station if feasible;
        // Recursive call of the Branch & Bound algorithm
        Next_Task ← Next task of Set_Of_Tasks;
        BB(Next_Task, Set_Of_Tasks);
        if no scalarization function improvement then
            Undo Current_Task movement;
        end
    end
end
return true if scalarization function is improved;
```

time limits of $c = 5755$ and $c = 7520$ (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 [1]. Originally, these instances were SALBP-1 instances[2] only having time information. However, we have created their area information by reverting the task graph to make them bi-objective (as done in [3]). The 9 TSALBP-1/3 instances considered are publicly available at `http://www.nissanchair.com/TSALBP`.

We run each algorithm 10 times with different random seeds, setting a fixed run time as stopping criterion (900 seconds). All the algorithms were launched in the same computer: Intel Pentium^{TM} D with two CPUs at 2.80GHz and CentOS Linux 4.0 as operating system. The specific parameter values considered for the different algorithms are shown in Table I. For the multiobjective local search, 20 as the maximum number of iterations and $MAX\_STATIONS = 20$.

B. Multiobjective Performance Indicators

We will consider two different multiobjective performance indicators [25], [26] to evaluate the quality of the memetic MACS proposal with respect to the TSALBP-1/3 state-of-the-art, the basic MACS algorithm and a GRASP method.

On the one hand, we selected one unary performance indicator: the hypervolume ratio (HVR). On the other hand,

[1]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.

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

TABLE I
USED PARAMETER VALUES FOR THE MULTIOBJECTIVE ALGORITHMS

Parameter	Value
MACS	
Number of ants	10
ρ	0.2
Ants' thresholds (2 ants per each)	$\{0.2, 0.4, 0.6, 0.7, 0.9\}$
β	2
q_0	0.2
GRASP	
γ	0.3
Diversity thresholds	$\{0.2, 0.4, 0.6, 0.7, 0.9\}$

we have also considered a binary performance indicator, the set coverage indicator C. We have used boxplots based on the C indicator that calculates the dominance degree of the approximate Pareto sets of every pair of algorithms (see Figure 1). Each rectangle contains nine boxplots representing the distribution of the C values for a certain ordered pair of algorithms in the nine problem instances (P1 to P9). Each box refers to algorithm A in the corresponding row and algorithm B in the corresponding column and gives the fraction of B covered by A $(C(A, B))$.

In addition, we use attainment surface plots [27] to ease the analysis of the results. The attainment surfaces plots of 4 problem instances, P1, P9, P8 and P3, appear in Figures 2 and 3.

C. Analysis of Results

The experimental results obtained by the memetic and basic MACS and the GRASP method can be seen in the C performance indicator boxplots of Figure 1, the HVR values in Table II (M-MACS corresponds to the memetic proposal and MACS to the basic one), and the attainment surfaces of Figures 2 and 3.

We can compare the behaviour of the memetic and basic MACS algorithms by analysing the C and HVR performance indicator values. The figures arise the following conclusions:

- The basic MACS algorithm is clearly outperformed by the memetic MACS variant. The difference is significant in view of the HVR values in Table II. There is a difference of about 0.2 between both algorithms. It means that the memetic MACS algorithm converges much more than the basic MACS.
- The C boxplots of Figure 1 are also untroubled. Almost all the solutions generated by the basic MACS algorithm are dominated by those obtained by the memetic MACS.

In addition, an analysis between the memetic MACS algorithm and the GRASP method is valuable. The HVR values and C boxplots of the GRASP method are also shown in Table II and Figure 1. The performance of the GRASP method is much higher than the basic MACS algorithm according to

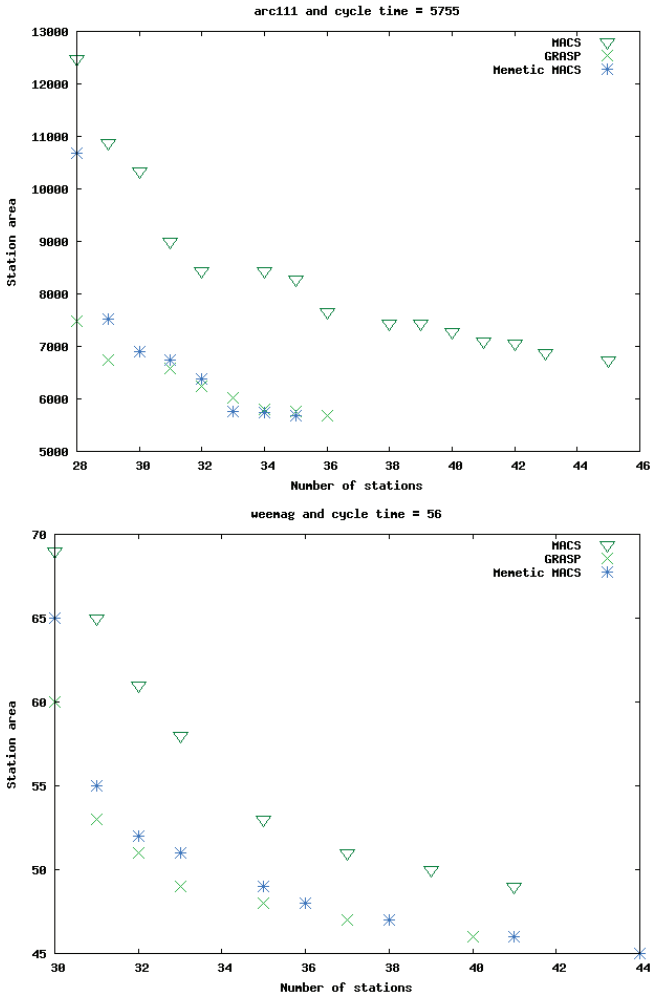

Fig. 2. Attainment surfaces for the P1 and P9 problem instances.

all the performance indicators. Therefore, we can state that the memetic algorithms outperform the basic MACS algorithm.

A comparison between the memetic MACS and GRASP is more difficult since their behaviour varies depending on the problem instance. Again, taking into account the C boxplots and HVR values, the memetic MACS algorithm performance is better than GRASP in P2, P3, P7, and P8, but worse in P1, P5, P6 and P9. In P4, P5, and P6, they behave similarly and the values of the performance indicators are very close. Therefore, it cannot be stated which of these two MAs is the best one without focusing on a particular instance.

Figures 2 and 3 graphically shows the aggregated Pareto fronts corresponding to P1, P9, P3, and P8 respectively. The same conclusions arise and the convergence differences can be observed. The Pareto fronts obtained by the basic MACS algorithm are far from the pseudo-optimal Pareto front in all the cases. The memetic MACS algorithm and the GRASP method converge much more.

The attainment surface plots also corroborate the fact that the behaviour of the memetic MACS and GRASP depends on the problem instance and we cannot assert which one is

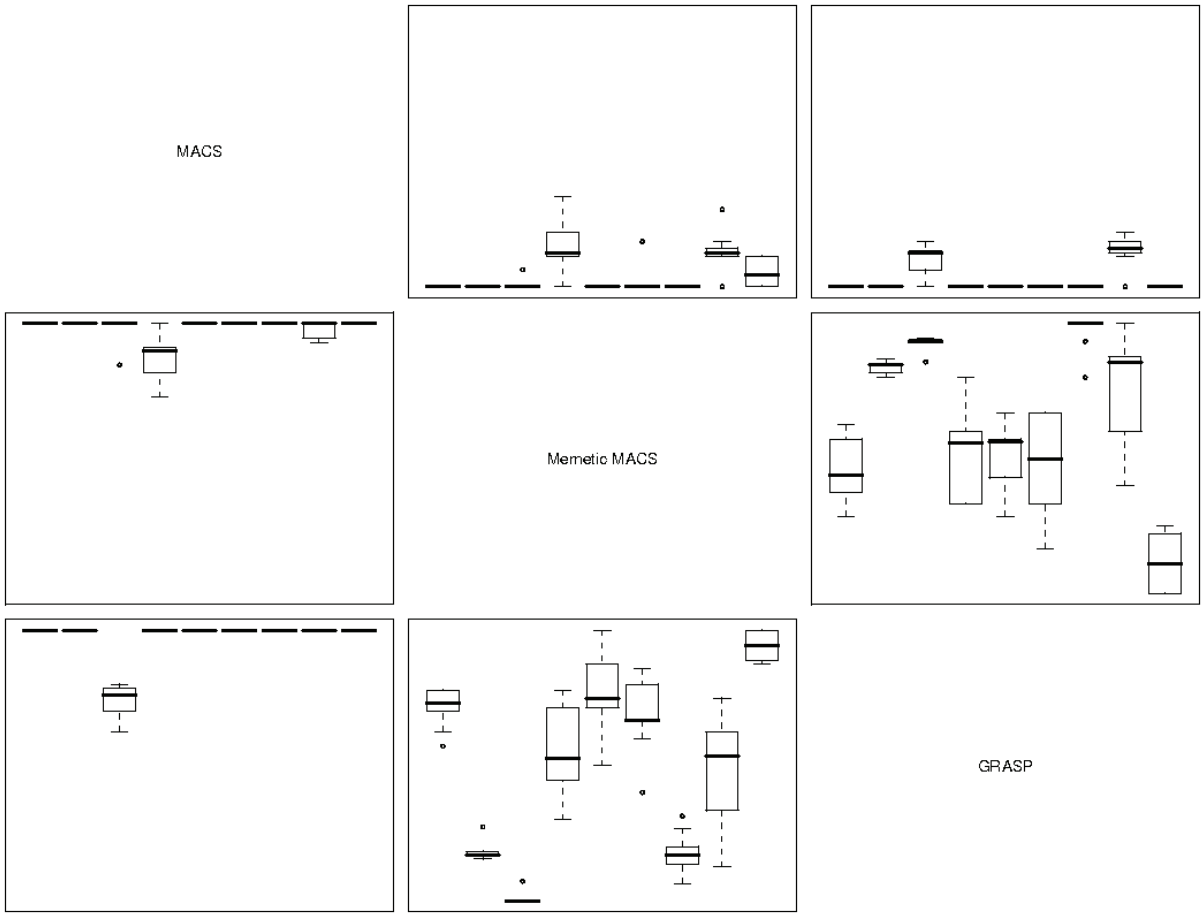

Fig. 1. C metric values represented by means of boxplots comparing the memetic MACS with the basic MACS and the GRASP method in the 9 problem instances.

TABLE II
MEAN AND STANDARD DEVIATION VALUES (IN BRACKETS) OF THE HVR METRIC FOR ALL THE PROBLEM INSTANCES.

	P1	P2	P3	P4	P5	P6	P7	P8	P9
MACS	0.776 (0.005)	0.77 (0.01)	0.707 (0.01)	0.719 (0.015)	0.622 (0.017)	0.611 (0.024)	0.725 (0.01)	0.619 (0.007)	0.758 (0.008)
M-MACS	0.967 (0.003)	**0.97 (0.003)**	**0.981 (0.002)**	0.946 (0.015)	0.916 (0.01)	0.949 (0.016)	**0.993 (0.001)**	**0.994 (0.003)**	0.965 (0.004)
GRASP	**0.994 (0.001)**	**0.97 (0.001)**	0.903 (0.003)	**0.984 (0.003)**	**0.941 (0.011)**	**0.973 (0.007)**	0.981 (0.002)	0.983 (0.003)	**0.988 (0.004)**

the best algorithm for the TSALBP. The memetic MACS algorithm is the best algorithm in P3 and P8 (Figure 3). However, the GRASP method is more suitable for P1 and P9 instance (Figure 2).

VI. CONCLUDING REMARKS

In this contribution, we have designed and applied a new memetic MACS algorithm to solve the TSALBP-1/3 in nine well-known problem instances. The new algorithm is multi-objective to tackle the industrial problem and makes use of a multiobjective local search procedure with two problem-specific local improvement methods, one per objective.

The memetic MACS algorithm shows a good behaviour in the majority of the problem instances, obtaining much better results than the state-of-the-art algorithm, MACS. The memetic MACS was also compared with a GRASP method. The best algorithm in quality is not clear enough since the memetic MACS and GRASP performed differently depending on the problem instance.

We aim to explore in future works the application of the local search to a multiobjective genetic algorithm to increase the quality of the Pareto fronts and try to obtain more diverse solutions. The application of the designed memetic approaches to real case studies is also planned.

ACKNOWLEDGEMENT

This work is supported by the UPC Nissan Chair and the Spanish Ministerio de Educación y Ciencia under project DPI2010-16759 (PROTHIUS-III) and by the Spanish Ministerio de Ciencia e Innovación under project TIN2009-07727,

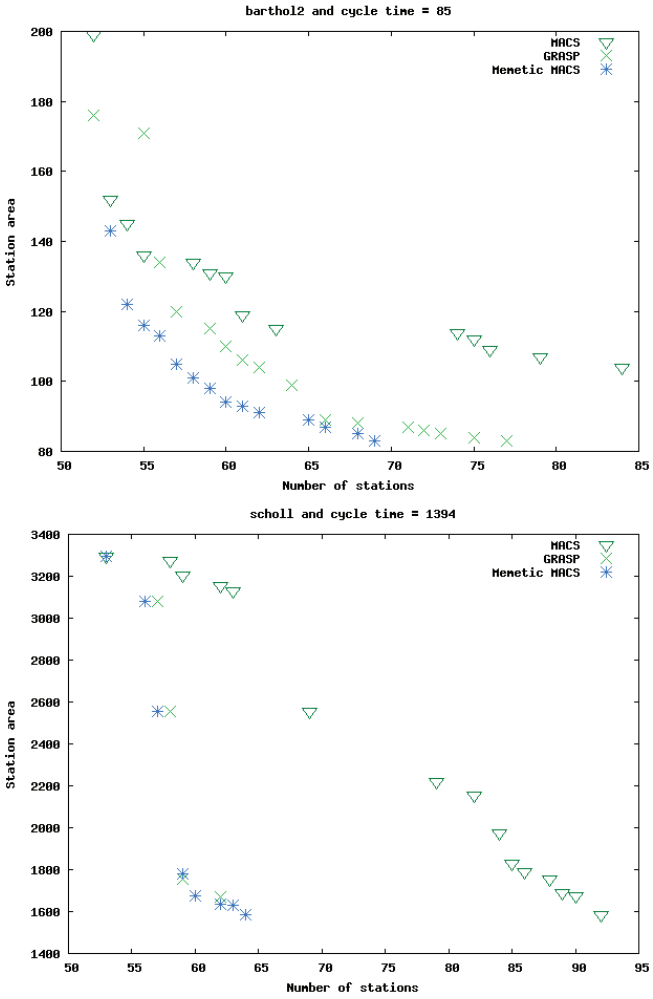

Fig. 3. Attainment surfaces for the P3 and P8 problem instances.

both including EDRF fundings.

REFERENCES

[1] I. Baybars, "A survey of exact algorithms for the simple assembly line balancing problem," *Management Science*, vol. 32, no. 8, pp. 909–932, 1986.

[2] A. Scholl, *Balancing and Sequencing of Assembly Lines (2nd. Edition)*. Physica-Verlag, Heidelberg, 1999.

[3] J. Bautista and J. Pereira, "Ant algorithms for a time and space constrained assembly line balancing problem," *European Journal of Operational Research*, vol. 177, pp. 2016–2032, 2007.

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

[5] F. Glover and G. A. Kochenberger, Eds., *Handbook of Metaheuristics*. Kluwer Academic, 2003.

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

[7] B. Barán and M. Schaerer, "A multiobjective ant colony system for vehicle routing problem with time windows," in *21st IASTED International Conference*, Innsbruck (Germany), February 2003, pp. 97–102.

[8] M. Chica, O. Cordón, S. Damas, and J. Bautista, "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*, June 2010, pp. 656–665.

[9] T. A. Feo and M. G. C. Resende, "Greedy randomized adaptive search procedures," *Journal of Global Optimization*, vol. 6, pp. 109–133, 1995.

[10] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms," Caltech Concurrent Computation Program, Pasadena, Tech. Rep. 826, 1989.

[11] H. Ishibuchi, T. Yoshida, and T. Murata, "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flow shop scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 204–223, 2003.

[12] Y. S. Ong, M. Lim, N. Zhu, and K. Wong, "Classification of adaptive memetic algorithms: a comparative study," *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, vol. 36, no. 1, pp. 141–152, 2006.

[13] C. Prins, "Two memetic algorithms for heterogeneous fleet vehicle routing problems," *Engineering Applications of Artificial Intelligence*, vol. 22, pp. 916–928, 2009.

[14] J. Santamaría, O. Cordón, S. Damas, J. M. García-Torres, and A. Quirin, "Performance evaluation of memetic approaches in 3D reconstruction of forensic objects," *Soft Computing*, vol. 13, no. 8-9, pp. 883–904, 2009.

[15] A. Scholl and C. Becker, "State-of-the-art exact and heuristic solution procedures for simple assembly line balancing," *European Journal of Operational Research*, vol. 168, no. 3, pp. 666–693, 2006.

[16] M. Dorigo and L. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

[17] J. Teghem and A. Jaszkiewicz, "Multiple objective metaheuristics for combinatorial optimization: A tutorial," in *Proceedings of the 4th Metaheuristic International Conference (MIC 2003)*, Kyoto (Japan), 2003, pp. 25–28.

[18] L. Paquete and T. Stützle, "A study of stochastic local search algorithms for the biobjective QAP with correlated flow matrices," *European Journal of Operational Research*, vol. 169, pp. 943–959, 2006.

[19] J. Knowles and D. Corne, "On metrics for comparing nondominated sets," in *Proceedings of Evolutionary Multi-criterion Optimization (EMO 2003)*, ser. Lecture Notes in Computer Science, vol. 2632. Berlin, Germany: Springer-Verlag, 2003, pp. 295–310.

[20] E. Zitzler and L. Thiele, "Multiobjective evolutionary algoritms: A comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[21] X. Gandibleux and A. Freville, "Tabu search based procedure for solving the 0-1 multiobjective knapsack problem: The two objectives case," *Journal of Heuristics*, vol. 6, no. 3, pp. 361–383, 2000.

[22] M. P. Hansen, "Tabu search for multiobjective optimization: MOTS," in *13th International Conference on Multiple Criteria Decision Making (MCDM'97)*, Cape Town, South Africa, 1997.

[23] A. Jaszkiewicz, "Genetic local search for multiple objective combinatorial optimization," *European Journal of Operational Research*, vol. 137, no. 1, pp. 50–71, 2002.

[24] R. Rachamadugu and B. Talbot, "Improving the equality of workload assignments in assembly lines," *International Journal of Production Research*, vol. 29, pp. 619–633, 1991.

[25] C. A. Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary Algorithms for Solving Multi-objective Problems (2nd edition)*. Springer, 2007.

[26] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. Grunert da Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.

[27] C. M. Fonseca and P. J. Fleming, "On the performance assessment and comparison of stochastic multiobjective optimizers," in *Proceedings of the 4th International Conference on Parallel Problem Solving from Nature (PPSN)*, ser. Lecture Notes in Computer Science, vol. 1141, Berlin, Germany, 1996, pp. 584–593.