A Multiobjective GRASP for the 1/3 Variant of the Time and Space Assembly Line Balancing Problem

M. Chica[1], O. Cordón[1], S. Damas[1], and J. Bautista[2,3,⋆]

[1] European Centre for Soft Computing, Mieres (Asturias), Spain
{manuel.chica,oscar.cordon,sergio.damas}@softcomputing.es
[2] Universitat Politècnica de Catalunya, Barcelona, Spain
{joaquin.bautista}@upc.edu
[3] Nissan Chair
http://www.nissanchair.com

Abstract. Time and space assembly line balancing considers realistic multiobjective versions of the classical assembly line balancing industrial problems, involving the joint optimisation 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 algorithm, based on the GRASP methodology, 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 our proposal is demonstrated by means of performance indicators in four problem instances and a real one from a Nissan factory.

1 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]. 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) [2]. 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 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

⋆ This work is supported by the UPC Nissan Chair and the Spanish Ministerio de Educación y Ciencia under project DPI2007-63026 and by the Spanish Ministerio de Ciencia e Innovación under project TIN2009-07727, both including EDRF fundings.

N. García-Pedrajas et al. (Eds.): IEA/AIE 2010, Part III, LNAI 6098, pp. 656–665, 2010.

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 [3]. In [4,5], we successfully tackled the TSALBP-1/3 by means of a specific procedure based on the Multiple Ant Colony System (MACS) algorithm [6], that approach is the state-of-the-art of TSALBP-1/3.

In addition, we proposed a multiobjective randomised greedy algorithm [4]. Although the performance of the MACS algorithm was better, the randomised greedy algorithm showed a good behaviour in some problems instances.

In this paper, we propose a multiobjective greedy randomised adaptive search procedure (GRASP) [7]. Its first stage corresponds to the randomised greedy construction presented in [4]. The second stage is based on a multiobjective local search with two improvement methods, one per objective. We consider two different GRASP approaches: the typical GRASP scheme and an alternative one, characterised by a random plus greedy construction. An experimentation is carried out in five problem instances, including a real-world one from the Nissan industry plant of Barcelona, Spain. Multiobjective performance indicators are used to analyse the behaviour of the algorithms.

The paper is structured as follows. In Section 2, the problem formulation is explained. Then, our new GRASP proposal to solve the problem is described in Section 3. The experimentation setup as well as the analysis of results are presented in Section 4. Finally, some concluding remarks and future research are discussed in Section 5.

2 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 [1] focuses on grouping tasks in workstations by an efficient and coherent way.

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 [2]. 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 optimisation 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:

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

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 [4].

The specialised literature includes a large variety of exact and heuristic problem-solving procedures as well as metaheuristics for solving the SALBP [8]. However, there are not many proposals for solving the multiobjective 1/3 variant of the TSALBP. The MACS algorithm is the state-of-the-art for TSALBP-1/3 and was presented in [4], where its performance was compared against a multiobjective extension of the SALBP genetic algorithm and a multiobjective randomised greedy algorithm.

3 Our Multiobjective GRASP Proposal

In this section, our multiobjective proposal, based on the GRASP methodology, is described. Following a GRASP approach, a solution is generated at each iteration. As our problem is multiobjective, the solution will be included in a multiobjective archive if it is not dominated. The algorithm finishes with a set of non-dominated solutions generated during all the iterations. Section 3.1 explains the generation of the greedy solutions. The multiobjective local search applied to those solutions is detailed in Section 3.2.

3.1 First Stage: Generation of Randomised Greedy Solutions

In the creation of greedy solutions during GRASP we introduce randomness in two processes. On the one hand, we allow the random selection of the next task among the best candidates. It will 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, respectively). In addition, η_j is also proportional to the ratio between the number of successors of task j and the maximum number of successors of any eligible task.

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 than q are selected to be in the restricted candidate list (RCL). In the former expression, γ is the diversification-intensification trade-off control parameter. When $\gamma = 1$ we find a completely random choice inducing the maximum possible diversification. In contrast, if $\gamma = 0$ the choice is close to a pure greedy decision, with a low diversification. Proceeding in this way, the RCL size is adaptive and variable, thus achieving a good diversification-intensification trade-off. 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.

We have also considered an alternative construction procedure, introduced in [9] as random plus greedy construction. We will call it GRASP RCL2. It first chooses candidates randomly. Then it evaluates each candidate according to a greedy function to make the greedy choice. GRASP RCL2 first constructs the restricted candidate list (RCL2) with a fraction β ($0 \leq \beta \leq 1$) of the elements, selected at random. Then, it evaluates all the elements in RCL2, computing the heuristic value η_j. The iteration of the algorithm finishes selecting the best one, i.e. the element j having the highest η_j value. After a preliminary study, we found that $\beta = 0.5$ was the best value for this parameter.

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

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. The final output consists of a Pareto set approximation composed of the non-dominated solutions.

3.2 Second Stage: Multiobjective Local Search

The second phase of the GRASP methodology consists of the improvement of each constructed solution considering a local search procedure. Mainly there are two stochastic local search approaches to multiobjective combinatorial optimisation problems [10]. 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 [11]. The second class is based on different scalarizations of the objective function vector [12]. We will use the second approach for our GRASP proposal. A weighted sum scalarization of the two objectives of our problem, A and m, are calculated by the following formula: $Min\ (\lambda^1 A + \lambda^2 m)$.

This will be the function to be optimised by the local search. The weight vector $\lambda = (\lambda^1, \lambda^2)$ is created at random for each greedy solution constructed in the first phase of the GRASP.

In addition, we will consider two different local search methods, one per objective. The algorithm will apply them to the greedy solution depending on the weight vector λ. If $\lambda^1 > \lambda^2$, the local search method for minimising the A objective will be followed. Otherwise, the local search method for m will be considered. If the selected local search method does not succeed minimising the weighted sum scalarization, the other method is then applied.

The local improvement procedures for balancing problems are based on moves [11]. Our local search methods are based on such moves of the tasks. To explain this procedure, it is necessary to define for each task j the first, ES_j, and last, LP_j, station where a task may be assigned according to the assignment of its immediate predecessors and successors. In general, a move (j, k_1, k_2) describes the movement of the task j from station k_1 to station k_2, where $k1 \neq k2$ and $k2 \in [ES_j, LP_j]$. The description of the two local search methods for objectives A and m follows:

- **Local search method for objective A.** The pseudo-code of the method is described in Algorithm 1. In this method, the neighbourhood of the solution is built by means of the explained task moves. The goal is to reduce the area occupied by the station with the highest area by moving tasks to other stations. This process of selecting the station with the highest area and removing tasks from it is repeated $MAX\_ITERATIONS$ times. After a preliminary study, $MAX\_ITERATIONS = 20$ was considered.
- **Local search method for objective m.** In this case, the objective is reducing the number of stations m. From the initial greedy solution, a neighbour is created by moving all the tasks from the station with the lowest number of tasks to other stations, keeping a feasible solution. The loop of the method is shown in Algorithm 2. Given a station to be removed, the algorithm uses a branch & bound function (`BB` function in the pseudo-code) to search for a feasible solution having the 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.

4 Experiments

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

4.1 Problem Instances and Parameter Values

Five problem instances with different features have been selected for the experimentation: `arc111` with cycle time limits of $c = 5755$ and $c = 7520$ (P1 and P2), `lutz2`

Algorithm 1. The pseudo-code of the local search method for the A objective.

```
1  while Iterations ≤ 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 improvement AND Tasks ≠ ∅ do
5          Task ← First element of Set_Of_Tasks ;
6          Find First_Station and Last_Station of Task;
7          while no scalarization improvement do
8              Possible_Station ← station with the lowest area
               ∈ [First_Station,Last_Station];
9              Move Task from Target_Station to Possible_Station;
10             if scalarization improvement then
11                 Make the movement permanent;
12             end
13         end
14         Remove Task from Set_Of_Tasks;
15     end
16     if Target_Station = ∅ then
17         Remove Target_Station;
18     end
19     Iterations ← Iterations +1;
20 end
21 return true if scalarization is improved;
```

Algorithm 2. The pseudo-code of the local search method for the m objective.

```
1  while Iterations ≤ MAX_ITERATIONS do
2      Stations ← Ascending sort with respect no. of tasks;
3      i ← 1;
4      while i ≤ MAX_STATIONS AND no scalarization 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 First_Station and Last_Station;
9          end
10         BB(First element of Set_Of_Tasks, Set_Of_Tasks);
11         if no scalarization improvement then
12             i ← i + 1;
13         end
14     end
15     Iterations ← Iterations +1;
16 end
17 return true if scalarization is improved;
```

(P3), mukherje (P4), and nissan (P5). Originally, these instances but nissan were SALBP-1 instances[1] only having time information. However, we have created their area information by reverting the task graph to make them bi-objective (as done in [2]). In addition, we have considered a real-world problem instance (P5) corresponding to the assembly process of the Nissan Pathfinder engine, assembled at the Nissan industrial plant in Barcelona (Spain) [2]. The five 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 the time as stopping criteria (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. GRASP and GRASP RCL2 were launched with $\alpha = 0.3$ and $\beta = 0.5$ respectively. For the multiobjective local search, 20 as the maximum number of iterations and $MAX\_STATIONS = 20$. On the other hand, for the MACS algorithm, we consider 10 different ants, $\beta = 2$, an evaporation rate $\rho = 0.2$, and a value of 0.2 for the transition rule parameter q_0. The MACS algorithm also uses two ants for each of the five ants' thresholds considered $\{0.2, 0.4, 0.6, 0.7, 0.9\}$ to make the algorithm multicolony.

4.2 Multiobjective Performance Indicators

We will consider two different multiobjective performance indicators [13] to evaluate the quality of the new GRASP proposal with respect to the TSALBP-1/3 state-of-the-art, the MACS algorithm.

On the one hand, we selected one unary performance indicator: the hypervolume ratio (HVR). On the other hand, 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 five boxplots representing the distribution of the C values for a certain ordered pair of algorithms in the five problem instances (P1 to P5). 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, in Figure 2 we show an example of the aggregated Pareto front approximation for instance P1 to allow a visual analysis of the results.

4.3 Analysis of Results

The experimental results obtained by the two multiobjective GRASP variants, i.e. GRASP and GRASP RCL2, and the MACS algorithm can be seen in the C performance indicator boxplots of Figure 1, the HVR values in Table 1, and the attainment surfaces of Figure 2.

Considering the C boxplots and only the first four problem instances, from P1 to P4, we can draw the following conclusions:

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

- If we compare MACS with GRASP, our new multiobjective proposal outperforms the state-of-the-art algorithm. Boxplots are clear and GRASP obtains better[2] Pareto sets than MACS in all the instances but P5 (nissan). As the last problem instance is different from the rest, it will be discussed later.
- The behaviour of GRASP RCL2 with respect to MACS is not as clear as the previous comparison. GRASP RCL2 dominates MACS in two instances, P1 and P4. However, the Pareto sets obtained by the MACS algorithm are better in the P2 and P3 instances. Therefore, both algorithms behave similarly and we cannot say which one is better just taking in mind the C performance indicator.
- Comparing both versions of GRASP in the C boxplots (Figure 1), it can be observed that the original one is significantly "better" than GRASP RCL2. This behaviour is common in all the problem instances.

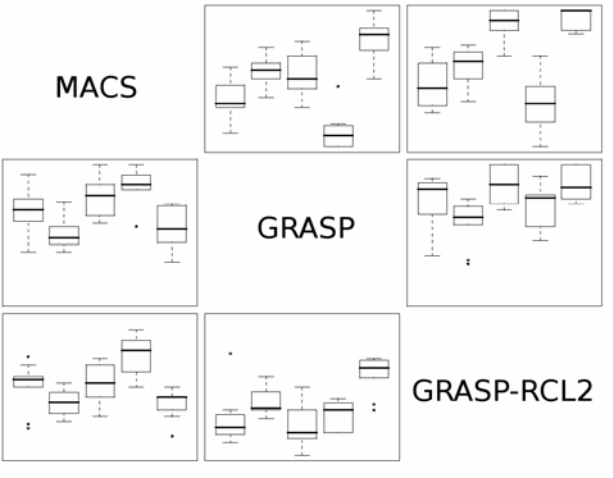

Fig. 1. C metric values represented by boxplots comparing MACS vs. the new GRASP proposals

Hence, according to the binary performance indicator C, the approach followed up by GRASP is useful to tackle the TSALBP-1/3. Nevertheless, the greedy construction phase is important for the problem solving since the difference between GRASP and GRASP RCL2 is high. Thus, selecting one task at random after restricting the candidate list is better than selecting the best task according to the greedy value after the random selection of the candidate tasks.

We can draw similar conclusions analysing the HVR values (see Table 1). The HVR values of GRASP are always the highest in all the problem instances but P5 (nissan). In contrast, if we compare the values of GRASP RCL2 and MACS, we can see how GRASP RCL2 is always better or equal than MACS. This means that generally

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

Table 1. Mean and standard deviation values (in brackets) of the HVR metric

	P1	P2	P3	P4	P5
MACS	0.925 (0.006)	0.914 (0.012)	0.877 (0.024)	0.886 (0.012)	**0.939 (0.009)**
GRASP	**0.969 (0.005)**	**0.935 (0.008)**	**0.889 (0.034)**	**0.944 (0.013)**	0.925 (0.027)
GRASP RCL2	0.948 (0.007)	0.919 (0.007)	0.827 (0.037)	0.909 (0.008)	0.896 (0.008)

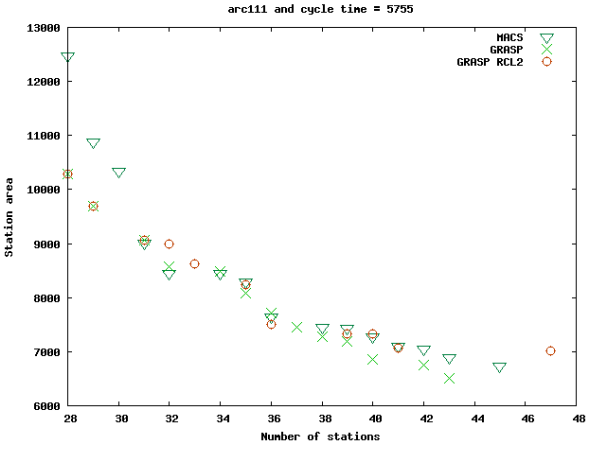

Fig. 2. Aggregated Pareto front approximation for the P1 problem instance

GRASP RCL2 converges better than MACS but its Pareto fronts are not as diverse as in MACS (the HVR values favour GRASP RCL2 against MACS).

As said, we have also considered the application of the algorithms to a real-world problem corresponding to the assembly process of the Nissan Pathfinder engine 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 Nissan instance data is also available at *http://www.nissanchair.com/TSALBP*.

With this instance, P5, results are different. If we analyse the C performance indicator (fifth column of the boxplots in Figure 1), MACS is better than both versions of GRASP. The corresponding values for the HVR performance indicator, Table 1, show the same behaviour, i.e. the MACS algorithm obtains Pareto sets of better quality.

Figure 2 graphically shows the aggregated Pareto fronts corresponding to P1 and P5. The same conclusions arise. GRASP is the best algorithm considering all instances but P5 (second Pareto front of the figure). However, MACS is more suitable for the Nissan problem instance, P5.

The P5 instance has some features making it different from the rest of the instances. As explained, it is the real case of the Nissan industry plant placed in Barcelona. The main difference of this problem instance with respect to the remainder is the area of the tasks to be allocated. Almost all the tasks have a very low area (less than one unit of area). Even more than 25 tasks have no area assigned. This particular characteristic

makes the MACS algorithm more competitive to solve this instance, making the Pareto set approximations spread out in a better way.

5 Concluding Remarks and Future Works

In this contribution, we have successfully applied a new algorithm based on the GRASP methodology to solve the TSALBP-1/3. The new algorithm is multiobjective to tackle the industrial problem and makes use of a multiobjective local search procedure with two problem-specific local improvement methods, one per objective.

Good results were achieved in the majority of the problem instances, obtaining even better results than the state-of-the-art algorithm, MACS. Nevertheless, the MACS algorithm still outperforms our multiobjective GRASP in the real-world Nissan instance. Therefore, we aim to explore in future works the application of the local search to the MACS algorithm as well as multiobjective memetic algorithms to increase the quality of the Pareto fronts.

References

1. Scholl, A.: Balancing and Sequencing of Assembly Lines, 2nd edn. Physica-Verlag, Heidelberg (1999)
2. Bautista, J., Pereira, J.: Ant algorithms for a time and space constrained assembly line balancing problem. European Journal of Operational Research 177, 2016–2032 (2007)
3. Glover, F., Kochenberger, G.A. (eds.): Handbook of Metaheuristics. Kluwer Academic, Dordrecht (2003)
4. Chica, M., Cordón, O., Damas, S., Bautista, J.: Multi-objective, constructive heuristics for the 1/3 variant of the time and space assembly line balancing problem: ACO and randomised greedy. Technical Report AFE-09-01, European Centre for Soft Computing, Asturias, Spain (2009) (submitted to Information Sciences)
5. Chica, M., Cordón, O., Damas, S., Bautista, J.: Adding diversity to a multiobjective ant colony algorithm for time and space assembly line balancing. In: 2009 IEEE International Symposium on Assembly and Manufacturing (ISAM 2009), Suwon, Korea, pp. 364–379 (2009)
6. Barán, B., Schaerer, M.: A multiobjective ant colony system for vehicle routing problem with time windows. In: 21st IASTED Conference, Innsbruck, Germany, pp. 97–102 (2003)
7. Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedures. Journal of Global Optimization 6, 109–133 (1995)
8. Scholl, A., Becker, C.: State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. European Journal of Operational Research 168(3), 666–693 (2006)
9. Resende, M.G.C., Werneck, R.F.: A hybrid heuristic for the p-median problem. Journal of Heuristics 10, 59–88 (2004)
10. Teghem, J., Jaszkiewicz, A.: Multiple objective metaheuristics for combinatorial optimization: A tutorial. In: 4th Metaheuristic International Conference (MIC 2003), Kyoto, Japan, pp. 25–28 (2003)
11. Rachamadugu, R., Talbot, B.: Improving the equality of workload assignments in assembly lines. International Journal of Production Research 29, 619–633 (1991)
12. Jaszkiewicz, A.: Genetic local search for multiple objective combinatorial optimization. European Journal of Operational Research 137(1), 50–71 (2002)
13. Coello, C.A., Lamont, G.B., Van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-objective Problems, 2nd edn. Springer, Heidelberg (2007)