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

M. Chica, O. Cordón, S. Damas
European Centre for Soft Computing
c/Gonzalo Gutiérrez Quirós, s/n
33600 Mieres (Asturias, Spain)
{manuel.chica, oscar.cordon, sergio.damas}
@softcomputing.es

J. Bautista, J. Pereira
Nissan Chair - UPC *
Av. Diagonal, 647
08028 Barcelona (Spain)
{joaquin.bautista, jorge.pereira}
@upc.edu

Abstract

We present a new multiobjective proposal based on Ant Colony Optimization to solve a more realistic extension of a classical industrial problem: Time and Space Assembly Line Balancing. Promising results are shown after applying the designed Multiobjective Ant Colony Optimization algorithm to four real-like problem instances.

Keywords: Ant Colony Optimization, Multiobjective optimization, Assembly Lines, Metaheuristics, Manufacturing, Production.

1 Introduction

The manufacturing of a production item is divided up into a set of tasks. Each task requires an operation time for its execution which is determined as a function of the manufacturing technologies and resources. A subset of tasks is assigned to each station of the plant. An usual and difficult problem is to determine how tasks can be assigned to the stations fulfilling the restrictions.

A family of this kind of problems is the one known as Simple Assembly Line Balancing Problem (SALBP). Taking this family as a base, Bautista et al. recently proposed a more realistic subfamily [2], Time and Space Assembly Line Balancing Problem (TSALBP), which considers an additional space constraint. This extended variant fits better to the real automotive industry scenario where our work is focused on.

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, their area and the number of the stations. In this paper we have selected the TSALBP-1/3 variant which tries to minimise the number of stations and their area for a given product cycle time. We have made this decision because it is quite realistic in the automotive industry. Moreover, the design of a multiobjective approach to solve it is one of the novelties presented in this contribution.

TSALBP-1/3 has an important set of constraints like precedences or cycle time limits for each station. Thus, the use of constructive metaheuristics like Ant Colony Optimization (ACO) [7] is more convenient than others like local or global search procedures [11].

Due to the two latter reasons, i.e., the multiobjective nature of the problem and the need to solve it through constructive algorithms, we will work with a multiobjective ACO (MOACO) algorithm [10]. This family involves different variants of ACO algorithms which aim to find not only one solution, but a set of the best solutions according to several conflicting objective functions. Pareto-based MOACO algorithms are included in this category and seem to be the most interesting.

We have chosen Multiple Ant Colony System (MACS) [1] to solve the TSALBP-1/3 because

* Universitat Politècnica de Catalunya

L. Magdalena, M. Ojeda-Aciego, J.L. Verdegay (eds): Proceedings of IPMU'08, pp. 1454–1461
Torremolinos (Málaga), June 22–27, 2008

of its good performance in the experimental study developed in [10]. Four real-like instances have been selected in order to test and analyse the performance of our new proposal.

This paper is structured as follows. In Section 2, the initial problem and all its variants are explained in depth. In Section 3, the details of the MOACO algorithm applied (MACS) are reviewed. In Section 4, a straightforward explanation of our approach for the general MACS scheme to solve the TSALBP-1/3 is tackled. Every experiment we did to check the performance of the algorithm besides their analysis are shown in Section 5. Finally, in Section 6, some concluding remarks and proposals for future work are commented on.

2 TSALBP and TSALBP-1/3

2.1 A brief introduction to the assembly line balancing problem

The manufacturing of a production item is divided up 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 its workload. A task j must be assigned to one 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.

In general, SALBP [16] focus on grouping together the tasks belonging to the set V in workstations by an efficient and coherent way. 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 on the stations. The literature includes a large variety of exact and heuristic problem-solving procedures as well as metaheuristics applied to the SALBP [15, 17].

2.2 TSALBP: the need of space constraints

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.

An area constraint may be considered by associating a required area a_j to each task j and by associating an available area A_k to each station k that, for the sake of simplicity, we shall assume to be identical for every station and equal to $A : A = max_{\forall k \in \{1..n\}}\{A_k\}$. Thus, each station k will require 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 that was 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:

1. all the precedence constraints are satisfied.
2. no station workload time, $t(S_k)$, is greater than the cycle time, c.
3. no area required by any station, $a(S_k)$, is greater than the available area per station, A.

TSALBP presents 8 variants depending on three optimization criteria: m (the number of stations), c (cycle time) and A (the area of the stations). Within these variants there are four multiobjective problems. For example, TSALBP-1/3 consists of minimising the number of stations m and the station area A, given a fixed value of the cycle time c.

Bautista et al. [2] proposed an ACO algorithm to solve a single-objective variant of the TSALBP. However, there is no approach in the literature to tackle any of the variants of TSALBP with a multiobjective approach.

We have chosen the 1/3 variant of TSALBP because it is quite realistic as the annual production of an industrial plant (and therefore, the cycle time c) is usually set by some market objectives. Besides, the search for the best number of stations and areas is logical if we want to reduce costs and make workers' day better by setting up less crowded stations.

3 Multiple Ant Colony System

As said, we have chosen MACS [1] because of its good performance in the experimental study developed in [10]. Moreover, for this initial work, we wanted a MOACO algorithm achieving a good convergence to all the Pareto front surface and not only to its central part or its extents as in other approaches [10].

MACS was proposed as a variation of the MACS-VRPTW [9], so it is also based on ACS [6]. Nevertheless, MACS uses a single pheromone trail matrix, τ, and several heuristic information functions, η_k, one per objective function, in our case η^0 and η^1. In this way, an ant moves from node i to node j by applying the following transition rule:

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

where β weights the relative importance of the objectives with respect to the pheromone trail, λ is computed for each ant h as $\lambda = h/M$, with M being the number of ants, and $\hat{i}$ is a node selected according to the following probability distribution:

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

Every time an ant crosses the edge a_{ij}, it performs the local pheromone update as follows:

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

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

$$\tau_0 = \frac{1}{\hat{f}^0 \,.\, \hat{f}^1}$$

However, the value of τ_0 is not fixed during the algorithm run, as usual in ACS, but it undergoes adaptation. At the end of each iteration, every complete solution built by the ants in the colony is compared to the Pareto set P generated till now to check if the former is a non-dominated solution. If so, it is included in the archive and all the solutions dominated by it are removed. Then, τ'_0 is calculated by applying the previous equation with the average values of each objective function taken from the solutions currently included in the Pareto set. Then, if $\tau'_0 > \tau_0$, the current initial pheromone value, the pheromone trails are reinitialised to the new value $\tau_0 \leftarrow \tau'_0$. Otherwise, the global update is performed with each solution S of the current Pareto optimal set P by applying the following rule on its composing edges a_{ij}:

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

4 Our approach to solve the TSALBP-1/3

4.1 Representation of a solution

In our case, one solution is an assignment of different tasks to different stations. The problem is that, contrary to other assignment problems like QAP [12], the number of stations is not fixed. Indeed, it is a variable to be minimised and we have to deal with the important issue of satisfying precedence constraints. As said, our procedure has to be constructive and it is important to decide

what kind of representation we have to consider. Using a constructive approach we can face the precedence problem. Likewise, an order codification may help us to give an adequate sequence of tasks as it is usually done for the SALBP [17]. Thus, our proposal will work with an order representation in which a sequence of tasks is represented.

However, giving a sequence does not solve the problem because the station assignment is still missing. Thus we have introduced the separator concept within the representation. Consequently, each position of our order scheme can be either a task number or a separator. We use separators to split up the sequence of tasks into different stations. A separator must be an integer (greater than the number of every task) to avoid the confusion with a task. This representation has been used in other problems like clustering [13, 14].

At the beginning, we decided to use a separator when the station was full in relation to the fixed cycle time c. We noticed that this scheme did not succeed because the Pareto front did not have enough diversity. Thus, we introduced a new mechanism in the construction algorithm to put a separator in accordance with a probability, given by the filling rate of the station:

$$p(sep) = \frac{\sum_{\forall i \in S_k} t_i}{c}$$

This probability threshold is computed at each construction step so it is progressively increasing its value. Once it has been computed, a random number is generated to decide if the station is closed using a separator or not with respect to that threshold.

4.2 Objective functions

According to the TSALBP formulation, the 1/3 variant manages with the optimization of the number of the stations, m, and the needed area, A. Using the notation given in Section 2, we can formulate its objectives as:

1. $Z_1(x) = m = \sum_{k=1}^{UB} max_{1 \leq j \leq n}\{x_{jk}\}$
2. $Z_2(x) = A = max_{1 \leq k \leq UB}\{\sum_{j=1}^{n} a_j x_{jk}\}$

4.3 Heuristic information

The multiobjective algorithm works with two different heuristic information values, each of them associated to one objective. The first one η_j^0 is related with the required time for each task and the second one η_j^1 with the required area:

$$\eta_j^0 = \frac{t_j}{UB_c} \times \frac{\mid F_j^* \mid}{max_{i \in V} \mid F_i^* \mid}$$

$$\eta_j^1 = \frac{a_j}{UB_A} \times \frac{\mid F_j^* \mid}{max_{i \in V} \mid F_i^* \mid}$$

where UB_A and UB_c are upper bounds for the area (the sum of all tasks' areas) and the time of each task (the given cycle time for the assembly line), respectively. All the heuristics give a value within the interval $[0, 1]$, with 1 being the most preferable.

Tasks having a large value of time (a large duration) and area (occupying a lot of space) are preferred to be firstly allocated in the stations. Regardless area and time information, we have added another information related to the number of succesors of the task which was already used in [2]. Tasks with a larger number of successors are preferred to be allocated first.

4.4 Pheromone trail and τ_0 calculation

The pheromone trail information has to memorise which tasks are the most appropiate to belong to a station. Hence, pheromone has to be associated to a pair $(task_{number}, station_{number})$. It is not useful depositing pheromone when we put a separator in our construction proccess because it is only a mark to distinguish one station from another.

In every ACO algorithm, an initial value for pheromone has to be set up. This value is called τ_0 and normally is obtained from an heuristic algorithm. We have used two single-objective greedy algorithms, one per heuristic, which work as follows. They open the

first station and select the best possible task according to their heuristic information (related either with the area and succesors, or the duration and succesors), repeating this process till no more tasks can be included due to the cycle time limit, so a new station will be opened. When no more tasks are to be assigned, the greedy algorithm finishes.

4.5 A basic random and an ACS algorithm

As there are not previous contributions to this problem, we are not able to compare our MACS approach against other proposals. So we have designed a basic multiobjective random search algorithm and a single-objective ACS algorithm [6] using the said coding.

The basic random search algorithm creates randomly a task sequence satisfying all precedence constraints. Starting with that sequence we need to divide it into stations fulfilling the cycle time limit for every station we create. To achieve that station assignment, the algorithm chooses one position to insert a separator at random but considering not to create a station with no tasks and not to exceed the cycle time limit. The algorithm finishes when all tasks are assigned to a station. The non-dominated solution archive and all the multiobjective mechanisms have been built as in the MACS algorithm.

On the other hand, the ACS algorithm has been designed according to the original version proposed in [6]. The most important issue is how to tackle a multi-objective problem designing a single-objective ACO algorithm. This problem has been resoluted by means of a weighted rule for the aggregation of the two objectives into one as follows:

$$Z(x) = \alpha \cdot m + (1 - \alpha) \cdot A$$

5 Experiments and analysis of results

This section shows the experimentation developed and a discussion on the results considering multiobjective metrics and some graphics.

5.1 Parameter values

The MOACO algorithm and the basic random search algorithm have been run ten times with ten different seeds for each of the four selected SALBP-1 instances [1]. The ACS algorithm has been run eleven times with different values of α parameter in order to spread all the extent of the Pareto front. The four problems have different features like time variability and order strength, and their names are arc111 (with cycle time $c = 5755$ and $c = 7520$), barthol2 ($c = 85$) and wee-mag ($c = 56$). Originally, these instances only have time information but we have created their area information from the latter by reverting the task graph to make them bi-objective (as done in [2]). All the considered parameter values are shown in Table 1.

Table 1: Used parameter values

Parameter	Value
Number of runs	10
Maximum run time	900 seconds
Number of ants	10
β	2
ρ	0.2
q_0	$\{0.2, 0.5, 0.8\}$
α for ACS	$\{0, 0.1, 0.2, ...0.9, 1\}$
PC Specs.	Intel Pentium^{TM} D 2 CPUs at 2.80GHz
OS	CentOS Linux 4.0 GCC 3.4.6

5.2 Results and analysis

The quality of our proposal is measured applying unary metrics: the number of total and different (in the objective vectors) Pareto solutions of each algorithm, and the S, $M2^*$ and $M3^*$ metrics [4]. S, the size of the space covered, measures the volume enclosed by the Pareto front, $M2^*$ evaluates the distribution of the solutions and $M3^*$ evaluates the extent of the Pareto fronts [2]. These metrics have been applied to the multiobjective algorithms.

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

[2] $M1^*$ has not been applied because we do not know the optimal Pareto fronts for this real-world problem.

Table 2: Metrics about the results of each algorithm

	# sols(σ)	# dif_sols(σ)	S(σ)	M2*(σ)	M3*(σ)
	ARC111 with cycle time = 5755				
Rand	11.1 (1.58)	10.9 (1.3)	9722162 (71239.19)	8.65 (0.79)	6303.02 (948.18)
M-0.2	9.7 (1.19)	9.6 (1.02)	10941016 (58244.8)	7.21 (0.98)	7951.11 (2401.1)
M-0.5	9.8 (0.98)	9.8 (0.98)	10920109 (36905.44)	6.95 (0.87)	7954.12 (3451.34)
M-0.8	8.7 (1.9)	8.6 (1.91)	10891762 (59504.72)	6.57 (1.1)	8199.41 (2791.27)
	ARC111 with cycle time = 7520				
Rand	11 (1.95)	11 (1.95)	11111291 (110417.72)	8.93 (1.27)	7028.72 (2109.64)
M-0.2	11.1 (1.76)	11.1 (1.76)	11824752 (75967.81)	8.02 (1.23)	9394.11 (3747.26)
M-0.5	9.9 (1.58)	9.9 (1.58)	11802513 (98203.98)	7.19 (0.8)	8169.41 (2018.62)
M-0.8	10.9 (2.17)	10.7 (1.9)	11754518 (68264.43)	8.29 (1.19)	7110.72 (1816.7)
	BARTHOL2 with cycle time = 85				
Rand	10.9 (1.97)	9.8 (1.72)	279722.31 (1893.13)	8.13 (1.29)	77.36 (25.17)
M-0.2	10.8 (1.94)	9.9 (1.04)	340742.5 (3055.09)	7.83 (1.1)	102.35 (37.96)
M-0.5	12.1 (3.14)	10.6 (2.29)	339581 (2608.08)	8.46 (1.38)	95.07 (36.63)
M-0.8	10.6 (2.87)	9.3 (1.79)	339321.59 (1937.75)	7.47 (1.41)	100.79 (32.8)
	WEEMAG with cycle time = 56				
Rand	16.3 (5.4)	8.2 (0.98)	50585.3 (431.63)	7.44 (0.73)	26.28 (2.02)
M-0.2	12.7 (4.71)	9 (1.48)	60890.7 (431.31)	7.99 (1.29)	32.15 (5.06)
M-0.5	11.6 (3.64)	8.3 (1.35)	60453.7 (566.7)	7.35 (1.28)	33.37 (14.57)
M-0.8	12 (2.93)	8.9 (1.04)	60188.9 (700.19)	8.11 (0.92)	28.32 (3.82)

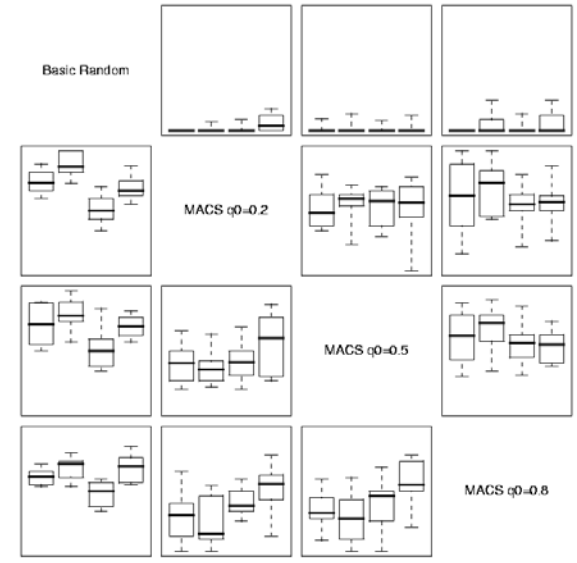

Figure 1: Box-plots of the results obtained in the C metric

In Table 2 we can observe the values of the different unary Pareto metrics. The MACS algorithm with $q_0 = 0.2$ achieves better and more vast Pareto fronts than the others in most of the cases. It obtains Pareto fronts with more diversity and better convergence than using higher values of q_0.

On the other hand, we have considered the binary metric C [4] to compare the obtained Pareto sets. Graphics in Figure 1 are box-plots based on C metric which compares the variants of the algorithm to each other by calculating the dominance degree of their respective Pareto sets. Each rectangle contains four box-plots (from left to right, arc111-5755, arc111-7560, barthol2 and wee-mag) representing the distribution of the C values for a certain ordered pair of algorithms. Each box refers to algorithm A associated with the corresponding row and algorithm B associated with the corresponding column and gives the fraction of B covered by A ($C(A, B)$). Consider, for instance, the top right box, which represents the fraction of solutions of MACS $q_0 = 0.8$ variant covered by the non-dominated sets produced by the basic random search algorithm.

Joining the box-plots (Figure 1) and the unary metrics (Table 2), we can reinforce that the $q_0 = 0.2$ variant is the best one, $q_0 = 0.5$ the second (with a short difference), and $q_0 = 0.8$ gets the worst results. As expected, all of them outperform the random search algorithm. Thus, a diversity scheme seems to be better than an intensification approach for the current problem. Nevertheless, in the fourth instance, wee-mag, the behaviour of the $q_0 = 0.8$ variant is not as bad as in the others. The reason is that in wee-mag every task is quite similar in terms of time and area, so the diversity is less important.

The graphical representation of the returned

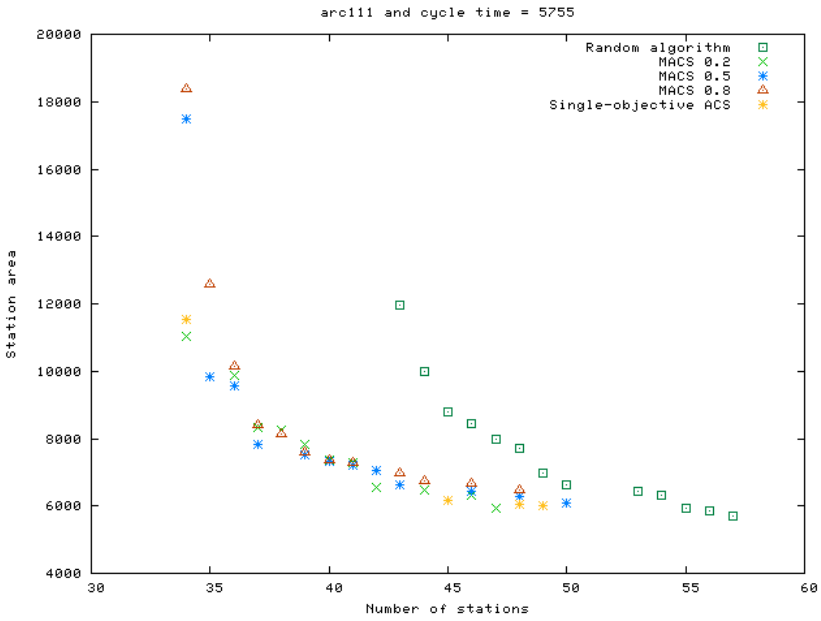

Figure 2: Pareto fronts for arc111 problem.

Pareto fronts for the arc111-5755 instance is shown in Figure 2. The four solutions obtained by joining the eleven runs of the ACS algorithm for this problem have been included as well. These graphic will help us to understand the values of the metrics. Because of space restrictions and pretty similar behaviours, we have only included the Pareto front of one problem instance.

In Figure 2, we can clearly see that all MACS variants outperform the basic random search but on the right-most extent of the Pareto fronts, where the largest exploration capability of the latter algorithm allows it to get a few additional solutions. This will be an issue to be tackled in future developments by increasing the diversity of MACS. Regardless the random search has not got as good convergence and diversity as MACS variants.

With respect to the ACS algorithm, every solution but one is dominated by MACS $q_0 = 0.2$. Besides, we can appreciate the good distribution of the Pareto front obtained by MACS $q_0 = 0.2$ in comparison with the formed ACS algorithm. Finally, we should remark that MACS $q_0 = 0.5$ obtains a good convergence to the central part of the front.

6 Concluding remarks and future work

In the current contribution we have proposed a new approach to tackle the TSALBP-1/3. The performance of a solution procedure based on the MACS algorithm with different parameter values has been presented and analysed. Bi-objective variants of four real-world assembly line problems have been used in our study.

From the obtained results we have found out that the yield of MACS algorithm is good to solve the problem, as Pareto sets of good quality with a number of different solutions have been achieved. Comparing the results of all MACS runs we notice that the algorithm works better when we use 0.2 as a value for q_0 parameter. Thus, there is a need to gain diversity in our problem to get better results.

Several ideas for future developments arise from this preliminary study: (i) performance of the considered MACS algorithm can be increased with some improvements like enhancing exploration using ants with different search behaviours (i.e., multi-colony); (ii) the comparison of the MOACO algorithm with more complex algorithms, for example, de-

signing a new GRASP [8] to solve it; (iii) selecting other MOACO algorithms in order to find out which the best one is; (iv) the solution of real-world multiobjective problem instances, above all, data from a real Nissan industry plant, placed in Barcelona (Spain).

Acknowledgements

We thank the UPC Nissan Chair as well as the Spanish Government for partially funding this work by means of a PROTHIUS-II project: DPI2007-63026.

References

[1] B. Barán, M. Schaerer. A multiobjective ant colony system for vehicle routing problem with time windows. In Proceedings of the 21st IASTED International Conference, pages 97-102, Innsbruck (Germany), February 2003.

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

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

[4] C.A. Coello, D.A. Van Veldhuizen, G.B. Lamont. Evolutionary Algorithms for Solving Multi-objective Problems. Kluwer, 2002.

[5] O. Cordón, F. Herrera, T. Stützle. A review on the ant colony optimization metaheuristic: Basis, models and new trends. Mathware and Soft Computing 9 (2-3), pages 141-175, 2002.

[6] M. Dorigo, L. Gambardella. Ant colony system: A cooperative learning approach to the travelling salesman problem. IEEE Transactions on Evolutionary Computation 1 (1), pages 53-66, 1997.

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

[8] T.A. Feo, M.G.C. Resende. Greedy Randomized Adaptive Search Procedures. Journal of Global Optimization 6, pages 109-134, 1995.

[9] L. Gambardella, E. Taillard, G. Agazzi. MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. In News ideas in optimization, pages 73-76, London, 1999.

[10] C. García Martínez, O. Cordón, F. Herrera. A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. European Journal of Operational Research 180, pages 116-148, 2007.

[11] G.A. Kochenberger, F. Glover. Handbook of Metaheuristics. Kluwer Academic, 2003.

[12] T.C. Koopmans, M.J. Beckmann. Assignment problems and the location of economics activities. Econometrica 25, pages 53-76, 1957.

[13] J.A. Lozano, P. Larrañaga, M. Graña. Partitional cluster analysis with genetic algorithms: searching for the number of clusters. In Data Science, Classification and Related Methods, pages 117-125. Springer, 1998.

[14] A.M. Robertson, P. Willett. Generation of Equifrequent Groups of Words Using a Genetic Algorithm. Journal of Documentation 50 (3), pages 213-232, 1994.

[15] A. Scholl, S. Voss. Simple assembly line balancing- Heuristic approaches. Journal of Heuristics 2, pages 217-244, 1996.

[16] A. Scholl. Balancing and sequencing of assembly lines. Physica-Verlag, Heidelberg, 1999.

[17] A. Scholl, C. Becker. State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. European Journal of Operational Research 180, pages 116-148, 2006.