Management and Production Engineering Review
Volume 6 • Number 1 • March 2015 • pp. 59–66
DOI: 10.1515/mper-2015-0008

AN ASSEMBLY LINE BALANCING PROBLEM AUTOMOTIVE CABLES

Hager Triki[1], Wafik Hachicha[2], Ahmed Mellouli[3], Faouzi Masmoudi[4]

[1] *University of Sfax, Sfax Engineering School, ENIS, Tunisia*
[2] *Higher Institute of Industrial Management Sfax, Tunisia*
[3] *University of Sousse, Sousse Engineering School, ENISo, Tunisia*
[4] *Laboratory Research of Mechanics, Modeling and Production (La2MP) Engineering School of Sfax, Tunisia*

Corresponding author:
Hager Triki
University of Sfax
Sfax Engineering School, ENIS, LA2MP:
Mechanic Modelisation and Production Laboratory Research
BP 1173, 3038, Sfax, Tunisia
phone: (+216) 97752848
e-mail: Hager_Triki@yahoo.fr

Received: 16 November 2014
Accepted: 11 January 2015

Abstract
In this paper, an Assembly Line Balancing Problem (ALBP) is presented in a real-world automotive cables manufacturer company. This company found it necessary to balance its line, since it needs to increase the production rate. In this ALBP, the number of stations is known and the objective is to minimize cycle time where both precedence and zoning constrains must be satisfied. This problem is formulated as a binary linear program (BLP). Since this problem is NP-hard, an innovative Genetic Algorithm (GA) is implemented. The full factorial design is used to obtain the better combination GA parameters and a simple convergence experimental study is performed on the stopping criteria to reduce computational time. Comparison of the proposed GA results with CPLEX software shows that, in a reasonable time, the GA generates consistent solutions that are very close to their optimal ones. Therefore, the proposed GA approach is very effective and competitive.

Keywords
assembly line balancing problem, genetic algorithm, cycle time, precedence and zoning constrains.

Introduction

The classical Simple Assembly Line Balancing Problem (SALBP) [1] has been widely enriched over the past few years with many realistic approaches and much effort has been made to reduce the distance between the academic theory and the industrial reality [2]. Each extension of SALBP is very useful for solving real practical problems to deal with many complicated constraints such as incompatibilities among tasks [3], space constrained [4] and the resource assignment [5] etc.

In this paper, the process of assembly line balancing of motor cables is studied in the multinational company LEONI (http://www.leoni.com) to increase the production rate. The company is a global supplier of wires, optical fibers, cables and cable systems as well as related development services for many applications in industries mainly in the automotive business.

In this industry, an interesting extension of SALBP is focused on, considering zoning constraints between sets of tasks, so that the incompatible tasks have to be assigned to different stations. For example, some raw materials (file, tube etc.) can be so similar to each other that quality consideration and the processing conditions force certain pairs of tasks to be assigned to different stations.

In this situation, the number of stations (m) is known and the objective is to minimize cycle time

where both precedence and zoning constrains between tasks must be satisfied. Since all versions of SALBPs are NP-hard [6], this ALBP is also NP-hard.

Many applications have been made in the literature to solve the ALBP using exact methods, such as linear programming [7], integer programming [8], dynamic programming [9] and branch-and-bound approaches [10]. However, these methods have proven to be effective only for small instances. Consequently, numerous researchers have been oriented towards the development of heuristics [11] and meta-heuristics such as tabu search [12], simulated annealing [13] and genetic algorithms [14]. The common characteristic of all the heuristic search methodologies is the use of problem-specific knowledge intelligently to reduce the search efforts [15]. Among these metaheuristics, genetic algorithms (GAs), which is received an increasing attention from the researchers since it provides an alternative to traditional optimization techniques by using directed random searches to locate optimum solutions in complex landscapes [16]. GA is a stochastic procedure which imitates the biological evolutionary process of genetic inheritance and the survival of the fittest. It is intelligent random search mechanisms for solving manufacturing optimization problems including scheduling problems, assembly line balancing and aggregate production planning. Those readers who are further interested in GA may refer to [17, 18] and [19]. Aytug et al. [19] reviewed over 110 papers using GAs to solve various types of production and operations management problems including control, facility layout design, line balancing, supply chain etc.

The existing studies in the literature have shown that GA method can be used as a very effective search technique in solving the NP-hard problems because it is able to move from one solution set to another and flexible to incorporate the problem specific characteristics. To reach these benefits, the standard GA design should be properly modified and adapted to the problem domain.

However, the majority of these previous studies have validated their GA performances using problem simulation, but little attention was paid to real case studies data [20, 21]. Furthermore, many studies have not integrated an optimization tool to choose their AG parameters.

Due to these limitations, an innovative genetic algorithm (GA) scheme is implemented in this paper. The proposed GA is designed in three stages. In the first stage, a new crossover operator is creed. In the second stage, a full factorial design is used to select the better GA parameters. In the third stage, a simple convergence experimental study is performed on the stopping criteria to reduce computational time. The evaluation of the effectiveness of this GA is realized through various sets of instances from a real case study of an automotive cable manufacturer.

The outline of the rest of this paper is as follows: Sec. 2 describes the problem and a proposed binary linear program (BLP) to solve this ALBP. Section 3 presents the steps of the proposed GA. Section 4 illustrates application case and the experimental results to select GA parameters. Section 5 shows computational results. Finally, concluding remarks are presented in Sec. 6.

Problem formulation

An instance (T, S, G) of the proposed ALBP consists of three components which are:

- $T = \{1...n\}$; a set of n tasks,
- $S = \{1...m\}$; a set of m stations,
- G; the precedence graph.

This ALB Problem (ALBP) has the following hypotheses and characteristics:

- The performance time of each operation is deterministic.
- The task cannot be subdivided.
- The precedence relationship among assembly tasks is known and invariable.
- A multiple product type is assembled on the line.
- No buffer is considered between the stations.
- A paced line with given number of stations (m).
- The zoning constraints are included in task assignment to stations.
- Every task is assigned to only one station.

The notation and decision variables considered in the mathematical model are defined as follows:

- Index and Parameters

i, j: The assembly tasks, $i, j = 1, 2, \ldots, n$
s: The stations, $s = 1, 2, \ldots, m$
n: Total number of assembly tasks
m: Total number of stations
t_i: Processing time of task i
pre(i): Set of direct predecessors of task i

- Decision variables

$$x_{js} \begin{cases} 1 & \text{If task } j \text{ is assigned to station } s \\ 0 & \text{otherwise} \end{cases}$$

IT: presents the set of incompatible pair tasks $(i, j) \in T$ which are incompatible (with $i < j$).

The problem can be formulated in the following way as a binary linear program (BLP) model:

$$\text{Min } C. \tag{1}$$

Subject to:

$$\sum_{s\in S} x_{is} = 1 \quad \forall i \in T, \tag{2}$$

$$\sum_{s\in S} sx_{is} \leq \sum_{s\in S} sx_{js} \quad \forall i,j \in T/i \in pre(j), \tag{3}$$

$$\sum_{i=1}^{n} t_i x_{is} \leq C \quad \forall s \in S, \tag{4}$$

$$x_{is} + x_{js} \leq 1 \quad \forall (i,j) \in IT, \quad s \in S, \tag{5}$$

$$x_{is} \in \{0,1\} \quad \forall s \in S \quad \forall i \in IT. \tag{6}$$

The objective function (1) minimises the cycle time C. Constraints (2) guarantee that every task $i \in T$ be assigned to only one station $s \in S$. Constraints (3) guarantee the respect of precedence relationships between the tasks. In case a task $j \in T$ is assigned to a station $s \in S$, all tasks $i \in pre(j)$ can only be assigned to stations $s' \in S$ with $s' \leq s$. Constraints (4) ensure that the sum of the processing times of the tasks assigned to a station $s \in S$ do not exceed the cycle time C. Inequalities (5) forbid the assignment of pair tasks of (IT) to the same station. Finally, constrains (6) ensure that x_{is} be a binary variable.

The presence of many constraints and binary variables makes mathematical resolution of large size problem, which is difficult to achieve with exact methods. Therefore, the proposed GA will be presented in detail the following sections.

The proposed Genetic Algorithm

Since the proposed problem is NP-hard, we chose to apply the GA method because it is a powerful tool used to found good solutions for NP hard problems.

Step 1: Chromosome coding

The encoding scheme is task-oriented and it is similar to the scheme generated by [22]. The length of the chromosome is equal to the number of tasks and each gene of the chromosome represents a task.

Step 2: Random generation of initial population

The initial population is randomly generated, assuring the feasibility of the precedence relations. These constraints may lead to produce unfeasible solutions. Elimination and replacement operations of these unfeasible solutions require an important time. In this paper, the method from [23] is applied to satisfy the precedence relations. Firstly, the tasks are registered based on their predecessors, and then a random task from the ones that have no predecessors (free tasks) is chosen to be assigned in the first sequence position of chromosome. Secondly, the set of free tasks is updated and the next task is selected from the new set. This procedure is repeated until the last task is assigned.

Step 3: Fitness evaluation

The fitness of a chromosome (individual) is defined as the inverse of the cycle time of the solution to be suitable with a minimisation problem. Therefore, the lowest cycle time the solution has, the more important its fitness is.

$f(j) = \dfrac{1}{C}$ denote the fitness function value of chromosome j; $j = 1, ..., J$.

The fitness value evaluates the individual performance in the search space.

The cycle time of the chromosome is determined by the following steps:

Step1: Determining the initial cycle time value which is Lower Bound (LB) calculated as follow:

$$LB = \max\left(\max_{1\leq i\leq n}\{t_i\}, \frac{\sum_{i=1}^{n} t_i}{m}\right). \tag{7}$$

Step2: Identifying the number of stations by the following procedure [24]: Tasks are assigned to stations according to the task sequence in the chromosome as long as the predetermined cycle time is not overtaken. Once this cycle time is surpassed at least for one model, or the zoning constraints of set IT are not satisfied, a new station is opened for assignment and the procedure is repeated until the last task is assigned.

Step 3: If the obtained number of stations is equal to a given m go to Step 4, otherwise the cycle time C will be incremented by 1 and go to Step 2.

Step 4: Finishing the procedure and presenting the cycle time of chromosome. Table 5 illustrates the steps of assignment of tasks to stations for an illustrative example of precedence graph G shown in Fig. 1 ($m = 6$ stations (S), $n = 10$ tasks, (4,5)$\in IT$).

Table 1
The assignment of tasks to stations of the illustrative example.

Step1	C=LB=8										
Chromosome associated to *G*		1	2	3	4	5	6	8	7	9	10
Step2	Task time	8	8	4	2	2	6	8	2	5	2
	Station	S1	S2	S3		S4		S5	S6		S7
Step3	C=LB=8 ⇨ *m* =7>6 ⇨ *C*=*C*+*1*= 9 go to step2										
Step2	Task time	8	8	4	2	2	6	8	2	5	2
	Station	S1	S2	S3		S4		S5	S6		
Step3	*C*=9 ⇨ *m* =6 ⇨ The desired number of stations										
Step4	End procedure: The cycle time of this chromosome is *C*=9										

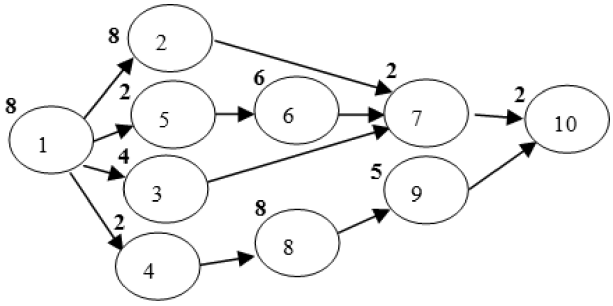

Fig. 1. Precedence graph G of the illustrative example.

Step 4: Reproduction: selection, crossover and mutation operator

- Selection operator

According to the reproduction probability of each individual in the population, the parent chromosomes are selected by the "Roulette Wheel strategy" proposed by [25].

- Crossover Operator (CO)

  – *First Crossover Operator* (*Two crossover points).*

The First Crossover Operator *(FCO)* was introduced by [26]. It works as follows:

*Two points randomly selected as indicated in Fig. 2, divide the parent into three parts (0-1, 1-2, 2-3).

*All elements from the parts (0-1, 2-3) of the first parent (P1) are copied to identical positions in the offspring1 (O1).

*All the elements within the part (1-2) of the first parent are reorganized according to the order of their appearance in the second parent vector (P2) to generate the remaining parts of offspring 1.

The other offspring (O2) is generated using the same method, with the roles of its parents reversed.

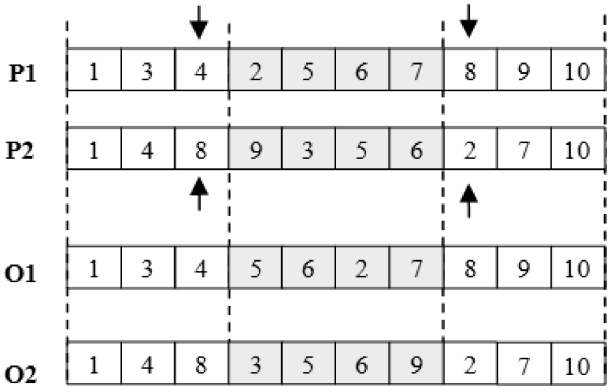

Fig. 2. The first crossover operator *(FCO).*

– *Second Crossover Operator* (*Four crossover points)*

The Second Crossover Operator *(SCO)* uses four crossover points, it works as follows:

*Four points generated randomly, divide each parent into five parts (0-1, 1-2, 2-3, 3-4, 4-5), as shown in Fig. 3.

*All elements from parts (0-1, 2-3, 4-5) of the first parent are copied to identical positions in the offspring 1.

*All the elements within the parts (1-2, 3-4) of the first parent are reorganized according to the order of their appearance in the second parent vector to generate the remaining parts of offspring 1.

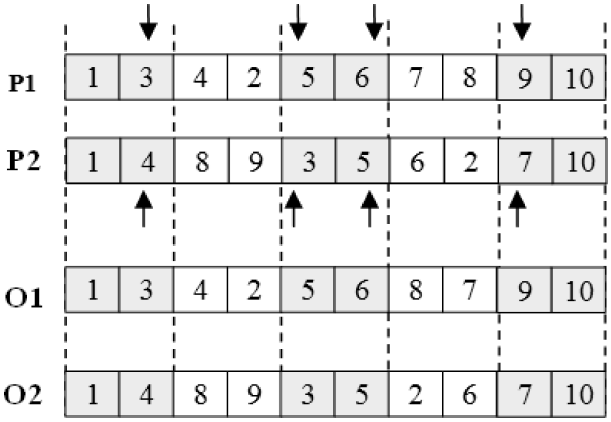

Fig. 3. The Second crossover operator *(SCO).*

- Mutation Operator: Scramble mutation

The procedure consists in randomly choosing one position within the solution offstring as it is shown in Fig. 4. This position cut the offspring into two parts (fragments). Fragment 1 and fragment 2 are presented as follows:

– Fragment 1: all elements are placed before the first position.
– Fragment 2: all elements are placed after the first position.

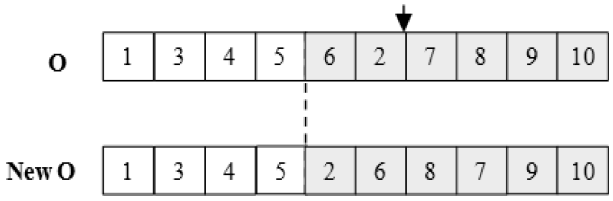

Fig. 4. Mutation Operator: Scramble mutation.

Fragment 1 is projected on the new mutated offspring (New O) but fragment 2 undergoes a reconstruction procedure [24].

Step 5: New population

A replacement strategy applied in [27] is used to create the new population. This strategy takes into account the fitness value of the individuals. The individuals of the new generation must have the best fitness (higher fitness value= minimum cycle time) of all individual forms:

i) the current population,
ii) off springs produced by crossover,
iii) off springs which underwent mutation.

The best solution of each generation is stored in order to avoid its loss over generations.

Step 6: Stopping criteria

The number of iterations is the stop condition. When this number is reached, the GA will be stopped otherwise returned to step 3.

Application case and analysis

The proposed GA was implemented with MATLAB 7.6. All experiments were run within Windows 7 Professional installed on a PC with Intel(R) Core (TM) i3-2310MCPU, 2.10 GHz. In order to characterize parameters and evaluate the performance of the GA, we explored some instances of an assembly line which produces many types of motor cable within the company LEONI.

Step 5: Selection of GA parameters

The performances of GA for both quality of solutions and compilation time are related to the different GA parameter values.

- Stopping criteria

The number of necessary iterations is not predefined. We have varied the stop number of iterations in order to minimise cycle time for some instances as indicated in Table 2.

Based on Table 2, the number of iterations should be fixed at $n \times 1000$ iterations for an initial population of 50 individuals. Indeed, it gives a minimum cycle time with an acceptable compilation time. Then the algorithm is stopped after $n \times 20$ generations $\left(\frac{n \times 1000}{\text{size of population}}\right)$ and returns the best solution.

Table 2
Evaluation of the cycle time of 3 instances for four numbers of iterations.

Number of iterations	Instance N°	The minimum cycle time	The maximum compilation time (minutes)
100n	1	70	6
	2	250	
	3	480	
500n	1	69	15
	2	240	
	3	470	
1000n	1	68	30
	2	230	
	3	475	
2000n	1	68	50
	2	230	
	3	475	

- GA parameters

The parameters of algorithm include population size (pop size), Crossover Operator (CO) and mutation rate (Pm). In order to select the better parameters combination, a full factorial design is used. For this reason, each parameter is divided with two levels: population size: 50 and 100, crossover operator: FCO and SCO and mutation rate: 0.1 and 0.15. There are three factors and two levels in the design of this experiment, and the target is to obtain the cycle time as minimum as possible. It follows that 8 (2^3) experiments are necessary to select the better configuration, as mentioned in Table 3.

Table 3
Tested GA parameters values.

Experiment	pop size	Crossover Operator	Pm
E1	0	0	1
E2	0	0	0
E3	0	1	1
E4	0	1	0
E5	1	0	1
E6	1	0	0
E7	1	1	1
E8	1	1	0

The result for all instances determined by Minitab14 is a same as shown in Fig. 5. Based in Fig. 5, the better combination of parameters consists in selecting: population size = 100, crossover operator = SCO and mutation rate = 0.15.

0: the first value mentioned of the parameter
1: the second value mentioned of the parameter

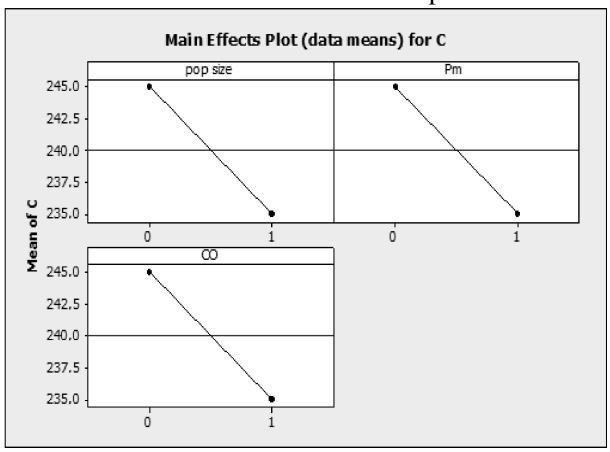

Fig. 5. Major influence diagram of cycle time C of instance 2.

We have found out that compilation time of GA is long. Consequently, we have modified once again the stopping criteria to decrease computation time. To determine the stopping criteria of GA, a simple convergence study is performed.

The evolution of the solution, as indicated in Fig. 6, is decomposed in two zones:

*Zone1: the quality of the solution improves through the number of generations.

*Zone 2: the quality of the solution is constant until the final generation (10n).

Then, in order to reduce computation time, the GA will stop when one of the following conditions is attained:

(i) The quality of solution is constant for 2n generations,

(ii) The total number of generations exceeds a maximum number 10n.

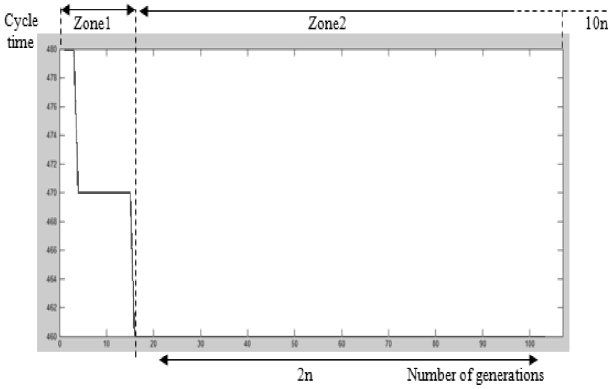

Fig. 6. Evolution of cycle time solution, as function of number of generations for instance 3 by the proposed GA.

The Fig. 7 records the reduction of the compilation time.

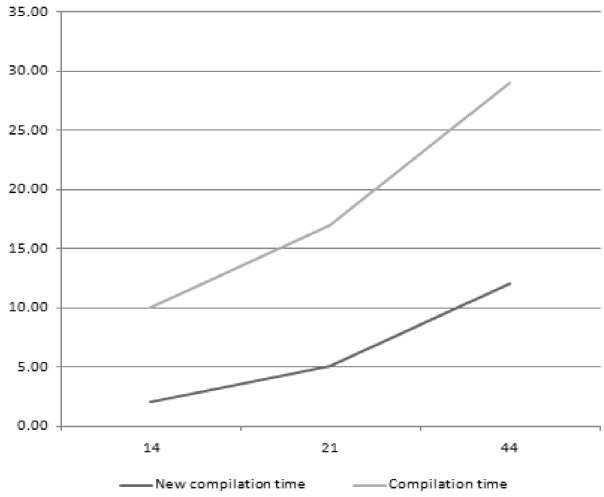

Fig. 7. Evolution of compilation time (CT), as function of number of tasks.

Computational results

Tables 4 and 5 contain the better solutions obtained by the CPLEX software (C^*) and by the proposed GA (C_{obtained}) for these real instances.

Table 4
The obtained solutions.

Problem	n	IT	m	C_{used} (in s)	C_{obtained} (in s)	C^* (in s)	E %	I %
1	14	4.6	3	69	67	67	0.00	2.9
2	21	1.7	5	220	220	220	0.00	0
3	44	1.17	6	480	460	443	3.83	4.17

C_{used}: Actual cycle time used in the line
C_{obtained}: Cycle time from the proposed GA
C^*: Optimal cycle time using CPLEX software

Table 5
The obtained compilation time.

Problem	CT1 (in min)	CT2 (in min)
1	2	30
2	5	65
3	12	122

CT1: Compilation time obtained by the proposed GA
CT2: Compilation time obtained by CPLEX software

Improvement rate

$$I\% = 100 \times \frac{C_{\text{used}} - C_{\text{obtained}}}{C_{\text{used}}}. \tag{8}$$

Percentage difference

$$E\% = 100 \times \frac{C_{\text{obtaiend}} - C^*}{C^*}. \tag{9}$$

As can be seen in the Tables 4 and 5, the proposed algorithm could make cycle time more efficient in all instances. The first comparison between the results given by the genetic approach with the actual cycle times used exhibits the reduction of the cycle time by 4% for a real assembly line. Then, the proposed GA approach can make a significant improvement to the performance of the company lines.

The second comparison of the results given by the genetic approach with the optimal cycle times shows the following points:

- The difference of two compilation times is very remarkable for all instances. The running time of CPLEX software is more important than the proposed GA. Therefore, the compilation time increases enormously depending on the problem size. In fact, CPLEX software loses its computational efficiency for large size problems.
- The E% does not exceed a rate of 4%.

Then, the second comparison of the results given by the genetic approach with the optimal cycle times shows an accomplishment of this algorithm to generate solutions very close from the optimal ones in a reasonable time (few minutes).

Conclusion

In this paper, an interesting extension of SALBP was focused on within a company of an automotive cables manufacturing. For this purpose, an innovative Genetic Algorithm (GA) was implemented. The effectiveness of the proposed GA is evaluated through various sets of instances collected from a real case study of an automotive cable manufacturer. The results of comparative studies with CPLEX software exact solutions exhibit that the proposed GA approach is very effective and competitive.

Discussions with industrial engineers and the results obtained in the practise case suggest some research topics perspectives: (1) Taking the resource assignment problem into account, many new criteria (price, speed of resource, and so on) have to be considered. (2) Adding other objectives to the problem such as costs of resources.

References

[1] Baybars I., *A survey of exact algorithms for the simple assembly line balancing problem*, Manag. Sci., 21, 8, 909–932, 1986.

[2] Andre's C., Miralles C., Pastor R., *Balancing and scheduling tasks in assembly lines with sequence-dependent setup times*, European J. of Operat. Resear., 187, 3, 1212–122, 20083.

[3] Kyungchul P., Sungsoo P., Wanhee K., *A heuristic for an assembly line balancing problem with incompatibility, range, and partial precedence constraints*, Comput. Industrial Engine., 32, 2, 321–332, 1997.

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

[5] Gao J., Sun L., Wang L., Gen M., *An efficient approach for type II robotic assembly line balancing problems*, Comput. and Industr. Eng., 56, 3, 1065–1080, 2009.

[6] Gutjahr A.L., Nemhauser G.L., *An algorithm for the line balancing problem*, Managem. Sc., 11, 2, 308–15, 1964.

[7] Salveson M.E., *The assembly line balancing problem*, J. of Indus. Engine., 6, 3, 18–25, 1955.

[8] Bowman E.H., *Assembly line balancing by linear programming*, Operat Resear, 8, 385–389, 1960.

[9] Held M., Karp R.M., Shareshian R., *Assembly line balancing dynamic programming with precedence constraints*, Operat. Research, 11, 3, 442–459, 1963.

[10] Jackson J.R., *A computing procedure for a line balancing problem*, Manag. Science, 2, 3, 261–272, 1956.

[11] Dar-El E.M., Rubinovitch Y., *MUST-A multiple solutions technique for balancing single model assembly lines*, Manag. Scienc., 25, 11, 1105–1114, 1979.

[12] Peterson C., *A tabu search procedure for the simple assembly line balancing problem*, In the Proceedings of the Decision Science Institute Conference, Washington, DC, pp. 1502–1504, 1993.

[13] Suresh G., Sahu S., *Stochastic assembly line balancing using simulated annealing*, Int. J. of Prod. Research, 32, 8, 1801–1810, 1994.

[14] Falkenauer E., Delchambre A ., *A genetic algorithm for bin packing and line balancing*, In The Proceedings of the 1992 IEEE International Conference on Robotics and Automation, Nice, France, pp. 1189–1192, 1992.

[15] Sabuncuoglu I., Erel E., Tanyer M., *Assembly line balancing using genetic algorithms*, J. of Intel. Manufact., 11, 3, 295–310, 2000.

[16] Tasan S.O., Tunali S., *A review of the current applications of genetic algorithms in assembly line balancing*, J. Intel Manufactur., 19, 1, 49–69, 2008.

[17] Goldberg D.Eo., *Genetic Algorithms in Search. Optimization and Machine Learning*, Addison-Wesley, Reading, MA, 1989.

[18] Liepins G.E., Hilluard M.R., *Genetic algorithms: foundations and applications*, Annals of Opera. Res., 21, 1, 31–58, 1989.

[19] Aytug H., Khouja M., Vergara F.E., *Use of genetic algorithms to solve production and operations management problems: A review*, Int. J. of Product. Research, 41, 17, 3955–4009, 2003.

[20] Chan C.C.K., Hui P.C.L., Yeung K.W., Ng F.S.F., *Handling the assembly line balancing problem in the clothing industry using a genetic algorithm*, Int. J. of Clothing Science and Technol., 10, 1, 21–37, 1998.

[21] Valente S.A., Lopes H.S., Arruda L.V.R., *Genetic algorithms for the assembly line balancing problem: A real-world automotive application*, Soft Computing and Industry, R. Roy et al. [Eds.], pp. 319–327, 2002.

[22] Leu Y.Y., Matheson L.A., Rees L.P., *Assembly line balancing using genetic algorithms with heuristic generated initial populations and multiple criteria*, Decision Sciences, 15, 4, 581–606, 1994.

[23] Hamta N., Fatemi Ghomi S.M.T., Jolai F., Bahalke U., *Bi-criteria assembly line balancing by considering flexible operation times*, Appl. Math. Model, 35, 12, 5592–5608, 2011.

[24] Akpına S., MiracBayhan G., *A hybrid genetic algorithm for mixed model assembly line balancing problem with parallel workstations and zoning constraints*, Eng. Appl. of Artificial Intel, 24, 3, 449–457, 2011.

[25] Holland H.J., *Adaptation in natural and artificial systems*, Ann Arbor. Michigan: The University of Michigan Press, 1975.

[26] Rubinovitz J., Levitin G., *Genetic algorithm for assembly line balancing*, In J. of Production Economics, 41, 1–3, 343–354, 1995.

[27] Simaria A.S., Vilarinho P.M., *A genetic algorithm based approach to the mixed-model assembly line balancing problem of type II*, Comput. & Industr. Engin., 47, 4, 391–407, 2004.