International Journal of Business and Management Invention
ISSN (Online): 2319 – 8028, ISSN (Print): 2319 – 801X
www.ijbmi.org || Volume 6 Issue 5 || May. 2017 || PP—34-40

Solving Assembly Line Balancing Problem Using A Hybrid Genetic Algorithm With Zoning Constraints

Belassiria Imad[1], Mazouzi Mohamed[1], ELfezazi Said[2], Cherrafi Anass[2], ELMaskaoui Zakaria[1]

[1](*Mechanical department, ENSEM/ Hassan II University, Casablanca-Morocco)*
[2](*TA and QC Department, Cadi Ayyad University, Marrakech-Morocco)*

Abstract: *In this paper, we propose a hybrid genetic algorithm to solve assembly line balancing problem. We put into the optimization framework of maximizing assembly line efficiency and minimizing total idle time simultaneously. The model is able to deal with more realistic situation of assembly line balancing problem such as zoning constraints. The genetic algorithm may lack the capability of exploring the solution space effectively, so we aim to provide its exploring capability by sequentially hybridizing the well-known assignment rules heuristics with genetic algorithm.*

Keywords: *Assembly line balancing problem; genetic algorithm; hybrid; assembly line efficiency; assignment rules heuristics; zoning constraints.*

I. INTRODUCTION

Assembly lines have gained great importance in manufacturing of high quantity standardized products particularly automobile manufacturing [1]. An assembly line consists of a sequence of workstations in which components are consecutively added to create one semi-finished assembly, this one moves from one station to the next station until the final assembly is produced. In order to meet required performance, the assembly line needs to be balanced by assigning tasks to workstation in such a way that the assembly objective is fulfilled, the demand is met and the constraints are satisfied. Therefore, an assembly line balancing is an effective tool for improving productivity and increasing efficiency.

For simple assembly line balancing problems (SALBP), there are four types mostly treated [2], SALBP-I intends to assign tasks to workstations such that the number of stations is minimized for a pre-specified cycle time while SALBP-II aims to minimize the cycle time, or equivalently, maximizes the production rate for a specific number of stations. SALBP-E is the most general problem maximizing the line efficiency (E) thereby simultaneously minimizing c and m considering their interrelationship. The type F (SALBP-F) is a feasibility problem which is to establish whether or not a feasible line balance exists for a given combination of m and c. Those versions of the problems are NP-hard.

SALBP-I is the most frequently employed in literature and researchers have published many studies to deal with this kind of assembly line balancing problem. Bautista and Pereira [4] proposed a new procedure called bounded dynamic programming to calculate the number of tasks assigned to minimum number of workstations. Chiang et al. [5] developed a branch and bound algorithm to deal with multiple U-line balancing problem. Kilincci [6] presented a new heuristic algorithm based on Petri net approach which makes an order of firing sequence of transitions from Petri net model of precedence diagram. Thus, the tasks are assigned to workstations using this order.

Unlike SALBP-I, the objective of SALBP-II is to minimize cycle time with a predefined number of workstations. Gao et al. [7] used genetic algorithm hybridized with local search to solve robotic assembly line balancing problem, in which the assembly tasks has to be assigned to workstations, and each workstation needs to select one of the available robots to process the assigned tasks with the objective of minimum cycle time. Kim et al. [8] addressed two sided ALBP with the objective of minimizing cycle time for a fixed number of mated-stations, a genetic algorithm is also presented to efficiently solve this problem. Several other methods have been reported in the literature. For example, Miralles et al. [9] presented branch and bound for solving a new problem called assembly line worker assignment and balancing problem. Seyed-Alagheband et al. [10] developed a mathematical model and a novel simulated annealing algorithm for solving general ALBP type II.

Most researchers in literature deal with SALBP-I and SALBP-II, and the objective of maximize line efficiency (SALBP-E) is less used and more difficult to deal with because of its nonlinear form. In this paper we treat SALBP-E.

In the balancing of assembly lines, there are cases where task assignment restrictions are unavoidable. For example, there is a set of tasks requires an expensive resources; hence they can share the same workstation or a set of tasks requires different equipment, thereby they have to be assigned to different workstations. The

most of researchers address ALBPs by classical restrictions like precedence constraints which led to ignore many aspects of the real-world problem, in this case those solutions becomes unusable in the industry. In this paper, we take into consideration the zoning constraints which are typical in the automotive industry and thus more practical SALBP-E model with zoning constraints is addressed.

Solutions are often classified into two categories: exact or proximate. As SALBP-E is NP-hard and the required computational time for obtaining an optimal solution with an exact method for most of line balancing problems increases exponentially with the size of instance considered, as a consequence, approximate methods are clearly needed in order to cope with large scale cases [11]. Approximate methods can be divided into: bounded exact method, simple heuristics, and metaheuristics. In the large number of optimization problem, bounded exact method and simple heuristics may lack the capability to find near optimal solution. Or, metaheuristics can be employed to efficiently solve large scale instances of more complex forms of the SALBP.

The most popular metaheuristic algorithm seems to be genetic algorithm, which is widely used in literature. Baykaso˘glu and Ozbakır [12] proposed a new multiple-rule-based genetic algorithm for balancing U-type assembly lines with stochastic task times. Hamta et al. [13] used genetic algorithm to deal with flexible time assembly line balancing problem, in which they have considered a kind of machines that can perform tasks in a range of time between lower and upper bounds. Purnomo et al. [14] presented genetic algorithm and first-fit rule to deal with more realistic situation of the two-sided assembly line problems. Often, the standard scheme of genetic algorithm is completed with local search. In this case, Akpinar and Bayhan [15] proposed a sequentially hybridizing three heuristic methods, Kilbridge&Wester heuristic, phase-I of Moodie& Young method, and ranked positional weight technique, with genetic algorithm to solve mixed model assembly line balancing problem of type I. Furthermore, the following researchers have used also genetic algorithm to solve different kind of assembly line balancing problem (Gao et al. [7]; Hwang and Katayama [16]; Hwang et al. [17]; Kazemi et al [18]; Kim et al. [8]; Kulak et al [19]; Moon et al [20]; Rabbani et al. [21]; Yu and Yin [22]; Zacharia and Nearchou [23]; Zhang and Gen [24]). In the real world line balancing an additional difficulty has presented. In order to improve the capability of genetic algorithm to explore effectively the solution space especially for a large size and complex assembly line balancing problem, we integrate heuristic method called task assignment heuristics which gives rich seeds to initial population and mixed with randomly generated seeds. Thus, we present a hybridization of Genetic algorithm and task assignment heuristics which consists in such a way that an individual represents one rule (heuristic) from the list of task assignment rules, see TABLE I, to be used to construct a feasible solution. In this paper, hybrid genetic algorithm (hGA) is proposed to deal with SALBP-E.

TABLE I. LIST OF TASK ASSIGNMENT RULES

Rule no.	Task assignment rules
1	Shortest Processing Time (SPT)
2	Longest Processing time (LPT)
3	Minimum Total Number of Successor Tasks (MiTNST)
4	Maximum Total Number of Successor Tasks (MaTNST)
5	Minimum Total Time of Successor Tasks (MiTTST)
6	Maximum Total Time of Successor Tasks (MaTNST)
7	Minimum Total Number of Predecessor Tasks (MiTNPT)
8	Maximum Total Number of Predecessor Tasks (MaTNPT)
9	Minimum Total Time of Predecessor Tasks (MiTTPT)
10	Maximum Total Time of Predecessor Tasks (MaTTPT)

II. FORMULATION OF THE SALBP-E

The SALBP usually takes into account two constraints (the precedence constraints and the cycle time). In industry there are more realistic situations which make the SALBP hardly applicable.

We tackle here some of additional constraints, which commonly has been used in real practices, in order to help line managers to deal with assignment difficulties. Zoning constraints contain two types positive zoning and negative zoning. Positive zoning means a set of tasks must be assigned into the same workstation while negative zoning means a set of tasks must be assigned into different workstations.

1. Notation

I: set of tasks; I = {1, 2, ….., i, …., n}.

J: set of workstation; J = {1, 2, …., j, …, m}.

C: cycle time.

t_i: processing time for task i.

s_j: total idle time at workstation j.

F_i: set of successors of task i.

P_0: set of tasks that has no immediate predecessors;

$P_0 = \{i \in I | P(i) = \emptyset\}$.

2. Mathematical model

Decision variables:

$$x_{ij} = \begin{cases} 1 & \text{if task i is assigned to workstation j} \\ 0 & \text{otherwise.} \end{cases}$$

The SALBP-E model combines the SALBP-I and SALBP-II which is defined as $E = \frac{t_{sum}}{m \times C}$, so the idle time is $(m \times C) - t_{sum}$ .The objective function of the problem is to maximize assembly line efficiency and simultaneously minimize the idle time, and furthermore it can be achieved by minimizing the number of workstations and cycle time.

$$WE = \frac{\sum_{i=1}^{n} t_i}{m \times C};\ i \in I \quad (1)$$

The assembly line efficiency is always in the value of (0, 1) and it is a maximization function. When the efficiency closer to 1, the assembly line will has better performance in which it has lower idle time.

$$\text{Max } F = WE \quad (2)$$

In addition to the cycle time and precedence constraints, we have considered additional constraints, in which we can find in realistic situations. So the constraints of the mathematical model are as follows:

Subject to

$$\sum_{j=1}^{m} x_{ij} = 1, \quad i \in I \quad (3)$$

$$\sum_{j=1}^{m} x_{bj} - \sum_{j=1}^{m} x_{aj} \leq 0 \quad a \in I, b \in F_a \quad (4)$$

$$\sum_{i=1}^{n} (t_i x_{ij}) + s_j \leq C \quad (5)$$

$$\sum_{j=1}^{m} x_{aj} - \sum_{j=1}^{m} x_{bj} = 0, \quad (a, b) \in ZP \quad (6)$$

$$x_{aj} - x_{bj} \leq 1, \quad (a, b) \in ZN, j = 1, \dots, m \quad (7)$$

$$x_{ij} \in \{0, 1\}, \forall i \in I, \forall j \in J \quad (8a)$$

$$s_j \geq 0, \forall j \in J \quad (8b)$$

Constraint (3) ensures that each task is assigned to only one workstation. Constraint (4) describes that the assignment of a task must be follow the precedence constraint, i.e. a task can only be assigned when its entire predecessors are finished. Constraint (5) ensures that the sum of processing time and idle time in a workstation must be smaller than or equal to the cycle time. Constraint (6) is compatibility zoning constraint, this type of constraints are normally related with the use of common equipment or tooling. For example, if two tasks need the same equipment or have similar processing conditions it is desirable to assign these tasks to the same workstation. Constraint (7) is incompatibility zoning constraint. These constraints are usually imposed when a set of tasks require different equipment, thus they cannot share the same workstation. Or when it is not possible to perform some tasks in the same workstation for safety reasons.Eqs. (8a)-(8b) define domain of the decision variables.

III. PROPOSED HYBRID GENETIC ALGORITHM (HGA)

Flow diagram of the proposed hGA is depicted in Fig. 1. Initialization of the hGA contains generating the initial population randomly and also including the solutions obtained by task assignment heuristics, After the initialization phase genetic operators are used to improve the solution quality.

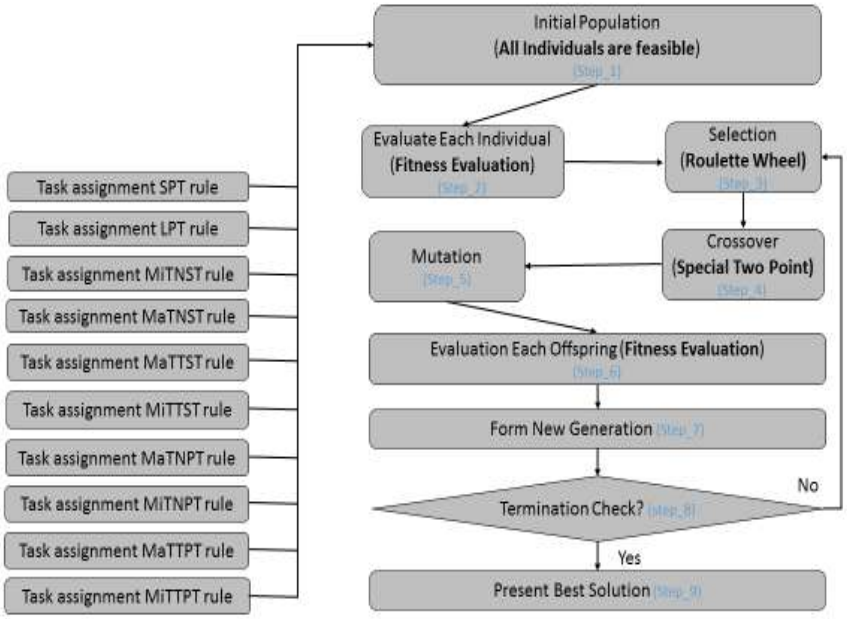

Fig. 1. Flow diagram of the proposed hybrid genetic algorithm

The proposed hGA is outlined step by step as follows:

Step-1: Generate an initial population of individuals obtained by the task assignment rules.

Step-2: Decode all individuals and evaluate the fitness function of their corresponding solutions.

Step-3: Determine pairs which define which chromosomes will undergo crossover (selection)

Step-4: Apply crossover operator to the selected pairs in order to obtain new pairs of chromosomes (offsprings).

Step-5: Apply mutation operator to one parent

Step-6: Decode all offspring and evaluate the fitness function of their corresponding solutions

Step-7: Decide which chromosome will form new generation; to preserve the diversity of the population, the offspring, which has not duplicated with any solution in the population, replace with the poor solution.

Step-8: If termination criterion is satisfied go to Step 9, else go to Step 3.

Step-9: Terminate the algorithm and present best solution.

1. Encoding and decoding

In the hGA, we used task based representation (Akpinar and Bayhan [15]). The length of the chromosome is defined by the number of tasks and each gene of the chromosome represents a task. Work assignment within the proposed approach is made by the following procedure. Tasks are assigned to workstations according to the task sequence in the chromosome, as long as the predetermined cycle time is not exceeded. Once the cycle time is exceeded at least for a model or the zoning constraints are not satisfied, a new workstation is opened for assignment and the procedure is repeated. Fig.2 illustrates assignment of tasks to workstations according to a chromosome.

2. Initial population

It is generally known that population diversity tends to show better performance. And using only the randomly generated initial population for GA may contain individuals from the same region of the solution space. Hence, using more heuristics to obtain initial solutions will give us chance to start the search from different regions of the solution space, and consequently to find good solutions in a reasonable amount of time [14].

To generate individuals in the initial population task assignment heuristics method, the following procedure is iterated.

Step 1: All tasks that have not been assigned are idendified. Find a set of tasks, F, such that all their predecessors have been assigned..

Step 2: Selecting a task i in F based on the selected task assignment rule. If there are more than one task with the same priority, a task is selected randomly.

Step 3: checking all the task constraints.

Step 4: If all constraints are fulfilled, task i is assigned to the workstation and then go to step 5. Otherwise, remove the task from F, and then go to step 2.

Step 5: If all tasks have been assigned, then stop. Otherwise, go to step 1.

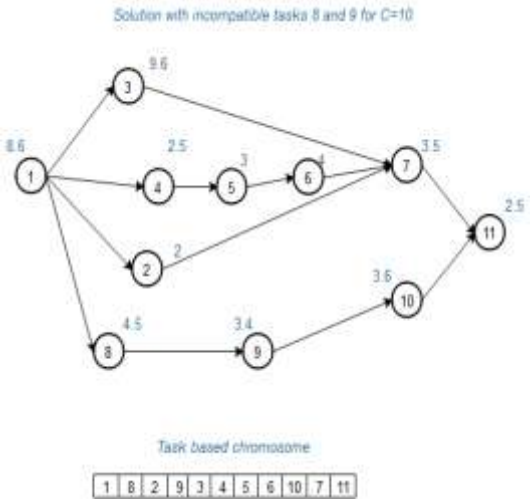

Fig. 2. Assignment procedure according to chromosome

3. Genetic operators

a. Order one crossover

In the order one crossover, one point, which cut each of the parents into two parts, is generated randomly. The idea is to preserve the relative order in which elements occur. The procedure of the crossover operation is described as follows.

Step 1: One integer value r between [1, n-1] are generated randomly.

Step 2: The genes with values of [1, r] are copied into the same position in child.

Step 3: The alteration procedure is used to fill the second part in O1.

Step 3.1: Identifying all tasks that have not been assigned in child, UT.

Step 3.2: Starting right from the cut point of the copied part.

Step 3.3: Select task i in UT, using the order of the second parent.

Step 3.4:Wrapping around at the end.

Step 4: Analogous for the second child, with parent roles reversed.

Assume that r=5; an example of order one crossover is illustrated in Fig.3.

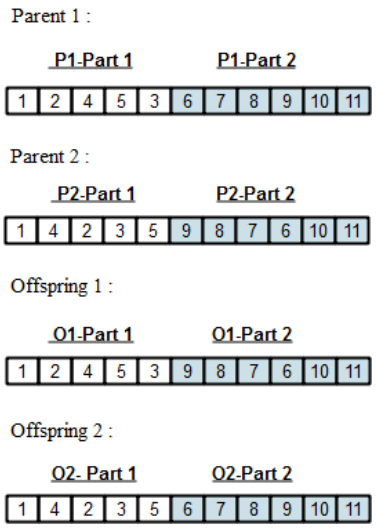

Fig. 3. Recombination: two point crossover

b. Scramble mutation

In the scramble mutation (Akpinar and Bayhan [14]), shown in Fig. 4., the procedure is described as follows. First, the point where the mutation occurs is decided randomly. Next, the head from the chosen parent is placed on the new mutated child. Then the tail of the new child is reconstructed using the procedure employed to generate initial population to again guarantee feasibility. This done by removing all references to head tasks in the prohibit table, and then randomly choosing a task from those in the table with no predecessor requirements. This new task is then added to the next locus in the chromosome and is removed from the prohibit table. The process continues until all tasks are assigned.

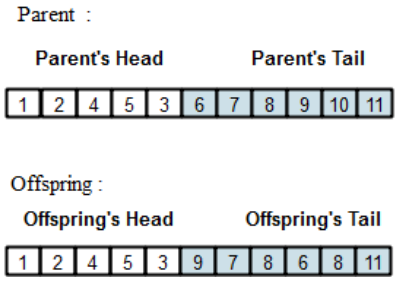

Fig. 4. Scramble mutation

IV. THE EXPERIMENTS AND RESULTS

In order to evaluate the performance of the proposed hGA in SALBP-E, the algorithm has been analyzed by running it on a set of real-world optimization problems [25]. Main characteristics of the benchmark data are exhibited in TABLE II. For each problem, we give the number of tasks N, the cycle time C and the minimal number of workstation m*. If m* is not equal to the theoretical minimum number of workstation mmin: = [tsum/C], the symbol '+' is added. The precedence graphs used for the problems set are shown in the fourth column. We classified the benchmark problems as medium-size (problems 1-3), and large-size (problems 4-10) according to the number of tasks they include. The algorithms are programmed in C# 6.0 (visual studio 2013), and the experiments are carried out on an EliteBook HP with 2.50 GHz i5-2520M CPU. The value of genetic parameters used for the hGA, crossover rate = 0.5 and mutation rate = 0.15 for all problems, for the population size we put Ip = 100. Since the size of solution space depends on the number of tasks, the termination criterion is differently set for each base problem. All the benchmark problems are also solved by pure GA and the task assignment heuristics used for sequential hybridization.

TABLE II exhibits results obtained for the benchmark problems by task assignment heuristics, pure GA and hGA in terms of number of workstation used (m) and the assembly line efficiency (WE%).

In this benchmark tests, we predefine a cycle time (C) for each problem, and we aim to minimize the total number of workstations (m), which is equivalent to maximize WE as mentioned in section II. We used these three measures in the current paper to be able to compare performance of our algorithm with those of others to find optimal solution for SALBP-E.

The experimental results illustrated in TABLE II show that for every experimental problem, hGA outperforms the task assignment heuristics to provide optimal number of workstations and maximum assembly line efficiency in all problems which conforms the ability of hGA to explore better solution than initial population. Compared with traditional GA, the result obtained in the experimental problems 1, 2, 4, 8, 9 by hGA and pure GA is compared. Otherwise, our hGA provide better assembly line efficiency than pure GA in the experimental problems 3, 5, 6, 7 and 10, which confirms their superior search capability to find higher quality solution and better convergence because of sequential hybridization with GA.

V. CONCLUSION

In the present work, the main goal is to sequentially hybridize the traditional GA with task assignment heuristics in order to improve its search ability to explore solution space and find higher quality solution and better convergence for SALBP-E with zoning constraints. Besides maximization of assembly line efficiency while simultaneously minimization of total idle time are considered. The hybridization is realized by inserting the solutions obtained by the assignment rules heuristics with randomly generated solutions as initial population of genetic algorithm. Thus, the solution quality of the pure GA was enhanced especially for large-size problems. For future research, search ability of the hGA can be extended by integrating specific heuristics like tree search, trade and transfer, random sampling, task grouping and approximation techniques.

TABLE II. COMPUTATIONAL RESULTS FOR REAL-WORLD PRECEDENCE GRAPHS FROM THE CURRENT BENCHMARK

	Problem N°	N	C	Graph	m*	Task assignment heuristics		Pure GA		Hybrid GA	
						m	WE (%)	m	WE (%)	m	WE (%)
Medium-size	1	28	216	Heskia	5	6	79,012	5	88,148	5	89,259
	2	32	2020	Lutz1	8^{+}	9	77,770	8	87,500	8	87,500
	3	35	54	Gunter	9	11	81,313	11	81,313	10	83,333
Large-size	4	53	2806	Hahn	6	7	71,400	6	83,310	6	83,310
	5	58	92	Warnecke	17	22	76,480	21	80,120	20	84,130
	6	70	320	Tonge	11	13	84,375	13	84,375	12	91,406
	7	89	20	Lutz2	25	29	83,620	28	86,660	27	89,815
	8	94	281	Mukherje	16^{+}	18	83,195	17	88,089	17	88,089
	9	148	564	Bartholdi	10	12	83,244	11	90,490	11	90,490
	10	297	1515	Scholl	46-47	53	86,951	52	88,623	51	90,360

REFERENCES

[1] Bagher, M., Zandieh, M. &Farsijani, H. Balancing of stochastic U-type assembly lines: an imperialist competitive algorithm. Int J AdvManufTechnol (2011) 54: 271.

[2] Christian Becker, Armin Scholl, A survey on problems and methods in generalized assembly line balancing, European Journal of Operational Research, Volume 168, Issue 3, 1 February 2006, Pages 694-715

[3] Sabuncuoglu, I., Erel, E. &Tanyer, M. Assembly line balancing using genetic algorithms. Journal of Intelligent Manufacturing (2000) 11: 295

[4] Joaquín Bautista, Jordi Pereira, A dynamic programming based heuristic for the assembly line balancing problem, European Journal of Operational Research, Volume 194, Issue 3, 1 May 2009, Pages 787-794.

[5] Chiang, W., Kouvelis, P., Urban, T., 2007. Line balancing in a just-in-time production environment: balancing multiple U-lines. IIE Transactions 39 (4), 347–359.

[6] Kilincci, O., 2011. Firing sequences backward algorithm for simple assembly line balancing problem of type 1. Computers & Industrial Engineering 60 (4), 830–839.

[7] Gao, J., Sun, L., Wang, L., Gen, M., 2009. An efficient approach for type II robotic assembly line balancing problems. Computers & Industrial Engineering 56 (3), 1065–1080.

[8] Kim, Y., Song, W., Kim, J., 2009. A mathematical model and a genetic algorithm for two-sided assembly line balancing. Computers and Operations Research 36 (3), 853–865.

[9] Miralles, C., Garcı´a-Sabater, J., Andre´ s, C., Cardos, M., 2008. Branch and bound procedures for solving the assembly line worker assignment and balancing problem: application to sheltered work centres for disabled. Discrete Applied Mathematics 156 (3), 352–367.

[10] Seyed-Alagheband, S., FatemiGhomi, S., Zandieh, M., 2011. A simulated annealing algorithm for balancing the assembly line type II problem with sequencedependent setup times between tasks. International Journal of Production Research 49 (3), 805–825.

[11] Battaïa, O., &Dolgui, A. (2013). A taxonomy of line balancing problems and their solutionapproaches. International Journal of Production Economics, 142(2), 259-277.

[12] Baykaso˘glu, A., Ozbakır, L., 2007. Stochastic u-line balancing using genetic algorithms. International Journal of Advanced Manufacturing Technology 32 (1), 139–147..

[13] Hamta, N., Ghomi, S. F., Jolai, F., &Bahalke, U. (2011). Bi-criteria assembly line balancing by considering flexible operation times. Applied Mathematical Modelling, 35(12), 5592-5608.

[14] Purnomo, H. D., Wee, H. M., & Rau, H. (2013). Two-sided assembly lines balancing with assignment restrictions. Mathematical and Computer Modelling, 57(1), 189-199.

[15] Akpınar, S., &Bayhan, G. M. (2011). A hybrid genetic algorithm for mixed model assembly line balancing problem with parallel workstations and zoning constraints. Engineering Applications of Artificial Intelligence, 24(3), 449-457.

[16] Hwang, R., Katayama, H., 2009. A multi-decision genetic approach for workload balancing of mixed-model U-shaped assembly line systems. International Journal of Production Research 47 (14), 3797–3822.

[17] Hwang, R., Katayama, H., Gen, M., 2008. U-shaped assembly line balancing problem with genetic algorithm. International Journal of Production Research 46 (16), 4637–4650.

[18] Kazemi, S., Ghodsi, R., Rabbani, M., Tavakkoli-Moghaddam, R., 2011. A novel twostage genetic algorithm for a mixed-model U-line balancing problem with duplicated tasks. International Journal of Advanced Manufacturing Technology 55 (9–12), 1111–1122.

[19] Kulak, O., Yilmaz, I., Gunther, H., 2008. A GA-based solution approach for balancing printed circuit board assembly lines. OR Spectrum 30 (3), 469–491.

[20] Moon, I., Logendran, R., Lee, J., 2009. Integrated assembly line balancing with resource restrictions. International Journal of Production Research 47 (19) 19, 5525–5541.

[21] Rabbani, M., Moghaddam, M., Manavizadeh, N., 2012. Balancing of mixed-model two-sided assembly lines with multiple U-shaped layout. International Journal of Advanced Manufacturing Technology 59 (9–12), 1191–1210.

[22] Yu, J., Yin, Y., 2010. Assembly line balancing based on an adaptive genetic algorithm. International Journal of Advanced Manufacturing Technology 48 (1), 347–354.

[23] Zacharia, P., Nearchou, A., 2012. Multi-objective fuzzy assembly line balancing using genetic algorithms. Journal of Intelligent Manufacturing, 23 (3), 615-627.

[24] Zhang, W., Gen, M., 2011. An efficient multiobjective genetic algorithm for mixedmodel assembly line balancing problem considering demand ratio-based cycle time. Journal of Intelligent Manufacturing 22 (3), 367–378.

[25] Scholl A. (1993) Data of Assembly Line Balancing Problems. SchriftenzurQuantitativenBetriebswirtschaftslehre 16, TH Darmstadt.