Genetic algorithm for type-2 assembly line balancing

Celso Gustavo Stall Sikora*, Thiago Cantos Lopes*, Heitor Silvério Lopes* and Leandro Magatão*
*Graduate Program in Electrical and Computer Engineering (CPGEI)
Federal University of Technology - Paraná (UTFPR), Curitiba, Paraná 80230-901
Email: magatao@utfpr.edu.br

***Abstract*—The assembly line balancing problem (ALBP) consists in finding the best assignment of tasks between several workstations. An evenly distribution reduces idle time and therefore results in more efficient production systems. Although several models have been proposed for ALBP, real lines present restrictions that usually violate simplifications assumptions. This paper presents a hybrid genetic algorithm to solve balancing problems with assignment restrictions. Heuristics are dynamically used in the encoding process to reduce search space and to focus the search on promising areas. The hybrid GA is able to obtain solutions close to the optimal (0.79% in average) for the most used dataset in the literature. The presented GA can incorporate equipment or zoning restrictions that might be present in real assembly lines.**

I. Introduction

The assembly line balancing problem (ALBP) is a classical problem in Operational Research. Its goal is to find an optimal distribution of tasks among workstations. Its first mathematical model was proposed by Salveson [1]. According to Baker [2], ALBPs are combinatorial NP-Hard problems.

An assembly line balancing problem restricted only by precedence relations between tasks (tasks that must be performed before other tasks) is called a simple assembly line balancing problem (SALBP). Precedence relations are often presented in the form of a precedence diagram such as the one presented by the Figure 1. The number of each task is given within each circle, while the numbers on the upper right corner of each node represent the duration of the task. In this we assume that the tasks are numbered following a topological ordering. SALBPs can also be classified into four categories, according to their goal [3].

SALBP-1 are problems in which a maximum cycle time is given and the goal is to minimize the number of workstations required to achieve a cycle time lesser or equal to the given maximum. SALBP-2 are problems in which the number of workstations is given and the goal is to minimize cycle time. When neither the cycle time nor the number of workstations is given, the problem is a SALBP-E, whose goal is to maximize the efficiency of the line by minimizing idle times. The last

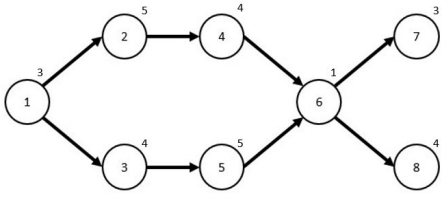

Fig. 1. Example of a precedence diagram.

variation is named SALBP-F, in which the goal is to reach a feasible solution for a given cycle time and a given number of workstations. According to Gamberini et al. [4], re-balancing problems related to working assembly lines are more frequent than planning the balancing of new assembly lines. Events such as a change in the production mix, the introduction of new product models, the increase or reduction of the production scale, amongst others, require a re-balancing of an assembly line [5]. Re-balancing problems usually present additional constrains, given that the assembly line is already present and certain alterations might be too expansive or outright unfeasible.

Many modifications of the ALBP have been proposed for the diverse particularities found. There are special models for U-shaped assembly lines [6], models for robotic lines [7], models for lines with assignment restrictions [8] and [9], amongst others. One important aspect of real assembly lines modeling is the capacity to translate features of the assembly line to the mathematical model. Boysen et al. [3] presents a survey on the many different features studied and described by different models.

On this work, a modeling approach employing a genetic algorithm is presented. The proposed assembly line balancing problem has as goal the minimization of the cycle time given a fixed number of stations (ALBP-2). However, in order to describe physical restrictions of the shop floor, assignment constrains are proposed. One example of physical restriction is a drilling task that can be only performed at stations geared with a drill; such restrictions create gaps of unfeasibility on the problem's domain. The problem contains still the time-wise precedence constrains between tasks, for instance, two pieces have to be assembled before they are wielded. Thus the search space is bounded by two types of restrictions.

The paper is organized as it follows: Section II presents the related works that treat ALBP with assignment restrictions. Section III describes the genetic algorithm proposed in the paper. Section IV describes the results obtained by the GA. The effect of each heuristic strategy is shown as well as a computational comparison with a highly specialized assembly line algorithm. The conclusion is presented in Section V.

II. Related Work

Out of SALBP's four variations, the SALBP-1 is the one most richly studied on the literature [10]. The algorithms that solve balancing problems with assignment constrains are usually designed for this variant of the problem.

Bautista and Pereira [8] introduced a new variation of such balancing problems: TSALBP (time and space constrained

978-1-4673-8418-6/15/$31.00 ©2015 IEEE

assembly line balancing problem). The author developed an Ant Colony Optimization Algorithm named ANTS for solving assembly line balancing problems in which each station was not only limited by the cycle-time, but also to a given amount of physical space that each task required. The space constraint reflects the workstation's needs for tools, equipment and pieces required by the tasks if they are to be performed at a given station.

Scholl et al. [9] introduces ABSALOM to solve ARALBP (assignment restricted assembly line balancing problem). The exact algorithm employs a Branch-and-Bound procedure to find solutions to the problem. On this problem's variations, not only time and space, but also any number of resources can be taken into account. Besides resources, allocations constraints are considered, such as linked tasks (which must be performed at the same station), incompatible tasks (which can not be performed at the same station) and restrictions on task-station allocation. ABSALOM can also solve TSALBP problems.

Sternatz [11] employs a multi-Hoffmann heuristic to solve assignment restricted problems. The procedure is capable of solving to optimality many instances of TSALBP and AR-ALBP with low processing time.

Akpınar and Bayhan [12] propose a hybrid genetic algorithm (hGA) for mixed model assembly lines with parallel workstations and zoning constraints. Three heuristics are used to provide good answers as initial solutions along with random individuals. The GA search focus on the neighborhoods of the heuristic solutions. The multiplicity of heuristics along with random individuals assures genetic variability.

The proposed algorithm differ from Akpınar and Bayan hGA [12] by using heuristics in the encoding of the genetic algorithm. The information of the best solutions are used dynamically to define how the genes are translated into a solution.

The assignment restrictions treated in our GA represent every unfeasible allocation of a task in a workstation. These restrictions can occur due to lack of the necessary equipment, lack of a skilled worker or other task-station incompatibilities within an assembly line. This approach suits well the re-balancing of assembly lines, where the workstations and equipment are already present and therefore imposes restrictions on the tasks' allocations.

The presented methods are all designed to solve ALBP-1: the cycle time is known and the goal is to minimize the number of stations. For ALBP-2, the majority of the algorithms employ a type-1 algorithm recursively for many trial cycle times. This allows a search for the minimum cycle time between a lower-bound and an upper-bound, that are determined by different heuristics.

Klein and Scholl [10] presented an algorithm to solve type-2 simple assembly line balancing problems (SALOME-2). It combines several heuristics to perform an informed laser search via branch-and-bound techniques. The authors also compare the efficiency of this algorithm with the SALBP-1 version (SALOME-1) employed repeatedly. In this paper, the algorithm is used as a comparison reference, since it is available on-line for download at http://www.assembly-line-balancing.de.

III. GA MODEL

The type-2 SALBP is a cycle time minimization problem. The problem is highly combinatorial, its variables represent allocation choices of each task. An assembly line with 50 tasks and 10 workstations, for instance, presents 10^{50} allocation options. Due to the problem's dimension and its combinatorial nature, a genetic algorithm is chosen as an appropriate way to model the problem.

According to a review of SALBP algorithms [13], the feasible solution generation affects greatly the algorithm's performance. Feasible solutions can be produced by either using rules for their generation, correcting solutions or applying punishments to unfeasible individuals. In the proposed GA, we use a codification that only allows feasible individuals.

A. Encoding

The most common GA codification strategies for ALBP are [14]: i) Workstation-oriented representation: there is one gene for each task. The gene stores the value of the workstation in which the task is allocated; ii) Sequence-oriented representation: the chromosome has one gene for each task. Each gene stores the value of a task. The sequence of the genes' value determines the allocations; iii) Partition-oriented representation: similar to the sequence-oriented representation, the genes store the sequence of tasks allocations. Besides that, the partition-oriented representation has genes to define the beginning and end positions of the allocations for each workstation. This paper uses an adapted workstation-oriented representation. Instead of storing the number of a workstation in a gene for each task, the GA's gene stores the position in a list of allocation possibilities. This way, both the physical allocation restrictions as the precedence restrictions are always satisfied by any codification. For example, one task can be allocated in workstations 2, 3 and 4. Thus, station 2 would be the first option of the list, station 3 would be the second and station 4 would occupy the third position.

The correspondence between the option codified by the genes and the workstation decision that it represents has to respect both precedence and physical allocation restrictions. Figure 2 illustrates how the precedence relations are used to update the list of possible allocations in order to allow only possible assignments. Figure 2 is based on the precedence graph showed in Figure 1. Before the first task has been allocated, when one consider only the precedence relations, any task can be assigned to any station. The algorithm associates tasks to stations in the order the tasks were given. In Figure 2.a, Task 1 is free to be allocated in any station (option 1 to 4). The gene information stores the value of the position of a list of possibilities. In the example, Task 1 is assigned to the first station, shown in green. Task 2 has to be allocated after Task 1. Once Task 1 is associated with the first workstation, Task 2 can be allocated to any workstation, including the first. Figure 2.b shows one example where Task 2 is assigned to the second position of the options list. This choice restricts Tasks 4, 6, 7 and 8 allocations in Workstation 1, once Task 2 precedes these tasks. The cells in black are used to illustrate the unfeasible allocations. Figures 2.c-f show the rest of the association process. The number of allocation options diminishes during the procedure, once the assignment of tasks

with precedence relation restricts the allocation possibilities of the next ones.

Physical allocation restrictions are used to model whether a task cannot be performed in a workstation. These restrictions do not depend on how the tasks are distributed, but they depend on the equipment, accessibility and compatibility of the stations. Figure 3 shows an example of a possibilities list under such restrictions. The list has to store only the feasible allocations.

Figure 4 illustrates the tasks association when both restrictions are applied simultaneously. The intersection of the two diagrams is significantly more restricted. The list of possible allocations is updated after each new allocation due to the precedence relations. This way, it is not necessary to apply corrections or punishments to obtain feasible solutions.

B. Variable set reduction

The precedence diagram can be used to restrict tasks allocation possibilities. For the problem presented by the diagram in Figure 1, for instance, the sum of the duration time of all tasks is 29. A task allocation with cycle time of 29 can be obtained by performing all tasks in one station. Any other allocation respects the limit of 29 time units. When one consider a cycle time of 28, not all of the task allocations are feasible. Once Task 1 have to precede all other tasks, it could not be allocated in the last station, as illustrated in Table I.

Gokcen and Erel [15] use the cycle time to restrain task allocations. They define a minimal and maximal station in which each task can be allocated. These limits are calculated with Equations (1) and (2). In the equations, t stands for duration time of Task i, P for the tasks that precede Task i, S for the tasks that succeed Task i, CT for the cycle time and NS for the number of stations.

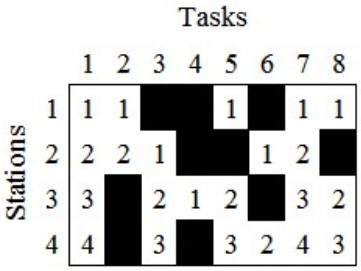

Fig. 3. Example of a list of possible allocations under physical allocation restrictions. The black cells represent infeasibilities.

$$S_{min_i} = \left[\frac{t_i + \sum_{j \in P} t_j}{CT}\right]^+ \quad (1)$$

$$S_{max_i} = NS + 1 - \left[\frac{t_i + \sum_{j \in S} t_j}{CT}\right]^+ \quad (2)$$

Different cycle times imply in different ranges of possible allocation for the tasks. The smaller the cycle time, the more restricted is the allocation interval. In the SALBP-1, cycle time is known, so the equations can be applied only once. In SALBP-2, the cycle time is the variable one wants to minimize. Therefore, in order to use equations (1) and (2), one needs a cycle time upper-bound. The closer this value is from the optimum value, the better the variable reduction is.

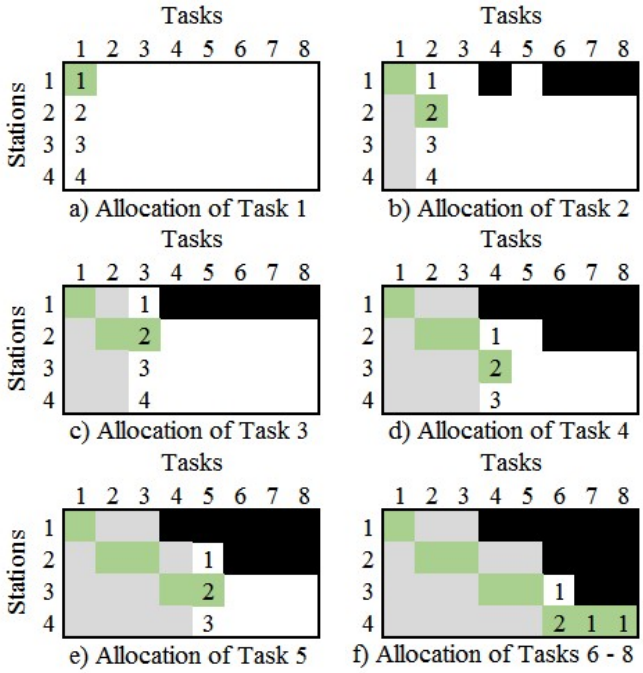

Fig. 2. Example of the allocation of tasks and the effect of precedence relations in the feasible allocation options.

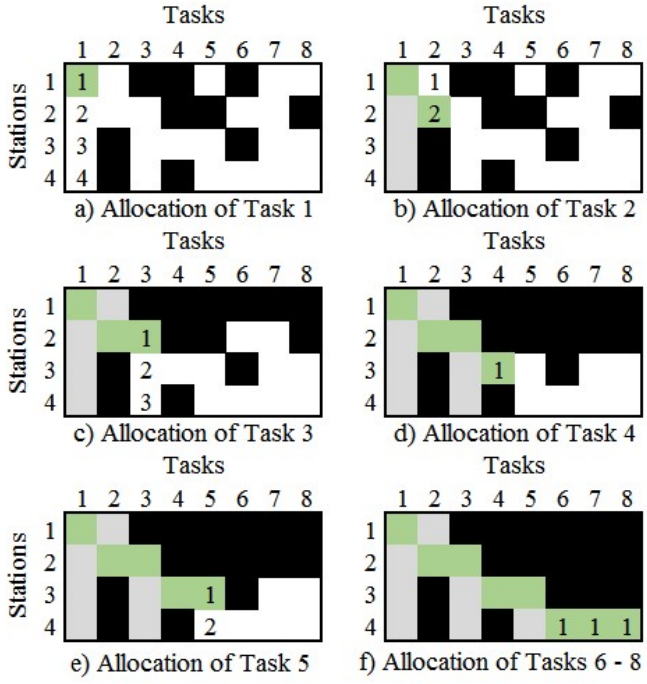

Fig. 4. Example of the allocation of tasks and the effect of precedence relations and physical restrictions in the feasible allocation options.

TABLE I. MINIMAL AND MAXIMAL STATIONS FOR CYCLE TIMES OF 8, 10, 12 AND 28 TIME UNITS.

Cycle Time	Bounds	Tasks 1	2	3	4	5	6	7	8
8	lower	1	1	1	2	2	3	4	4
	upper	1	2	2	3	3	4	4	4
10	lower	1	1	1	2	2	3	3	3
	upper	2	3	3	3	3	4	4	4
12	lower	1	1	1	1	2	2	3	3
	upper	2	3	3	4	4	4	4	4
28	lower	1	1	1	1	1	1	1	1
	upper	3	4	4	4	4	4	4	4

TABLE II. EXAMPLE OF THE ENCODING OF THE OPTION STORE BY THE GENE AND THE ASSOCIATED STATION.

Option - Gene	1	2	3	4	5	6	7	8	9	10
Station - Allocation			1			2		3	4	5

In this GA, the best solution found is used as a cycle time estimative to determinate the minimal and maximal station for each task. Every time a new better individual is found, the limits are updated, diminishing the search space for the individuals of the next generations. The minimal and maximal stations are used as restrictions to create the possible allocations list.

C. Search focus

In order to be able to create any possible solution combination, the list of options has to have at least the size of the number of stations. It is easy to see, however, that some allocations produce low fitness individuals. For instance, when a task with many successors is allocated in the end of the line, all tasks that depend on the first task have also to be allocated in the last station. If that happens, the workload at the end of the line would produce a bad quality balancing. Therefore, there is a natural tendency to allocate tasks sooner rather than later, for the next tasks would have more freedom to be allocated.

The GA uses this pattern to focus the search in more promising regions of the search domain. The option-station correspondence is not done in a uniform way, more than one position were used to represent the earlier stations. The quantity of options used to represent each station obeys an exponential distribution. The more options of stations an allocation enables, the more probable the allocation is. Table II exemplifies an asymmetric option-station correspondence. In this example, a task can be allocated in 5 stations, while there are 10 values that could be assumed by the gene. Once tasks are restricted by precedence relations, it is usually better to allocate them in the first stations.

The chromosome has a gene for each task. Each gene has a minimal number of possible values so that it can assume to an exponential value-station correspondence. Although the search space increases, using a gene with the number of possible values equal to ten times the number of stations showed the most promising results. The extra possibilities are used to fit the exponential distribution. Equation (3) shows the probability of each station to be chosen. The probability diminishes as the number of the latter stations (NS - i) decreases. The term b controls how focused the search is. When $b = 0$, the distribution is uniform, no focus is applied. For large values of b, the algorithm works almost as a greedy algorithm. S_f is the set of feasible station allocations.

$$P_i = \frac{exp[b(NS - i)]}{\sum_{j \in S_f} exp[b(NS - j)]} \quad (3)$$

The probability values have to be transformed into integer values. In order to not discard any allocation possibility, small probabilities are increased, so that there is at least one allocation for each one of the feasible stations. The rest of the values are determined according to the exponential distribution. The sum of the possibilities is equal to the range a gene can assume.

D. Loaded workstation closing

As an additional strategy, the concept of "closing" a workstation was employed. That is, cycle time is limited to a maximum value. Only allocations that do not load a workstation above the limit are considered. This way, the GA searches for unloaded stations to allocate tasks. The best cycle time found is used as the limit parameter. Each time a new best individual is found, the limit value is updated. The stations that are considered closed cannot receive more tasks. This restriction is implemented in the creation of the allocation possibilities list.

E. Implementation

The algorithm implementation is made using the software Gallops 3.2.2. The goal function of a SALBP-2 is the minimization of the cycle time given a number of workstations. The goal function is obtained by the maximum value of the sum of tasks' durations in each of the workstations. The better the tasks are distributed, the lower the cycle time will be. The fitness function is defined as the inverse of the cycle time.

The uniform crossover (90%) and the individual bit mutation (5% per bit) are the operators used. The GA uses 200 individuals that evolve during 1000 generations. The selection method used is the stochastic tourney of size 10. The used value for the exponential distribution parameter b is 1.4.

IV. RESULTS

The GA is tested with a SALBP-2 dataset. Scholl and Klein [16] uses a benchmark of 302 instances of SALBP problems with 29 to 297 tasks. This testbed is used as reference to compare algorithms. The search space of an ALBP can be calculated multiplying the amount of allocation options each task have. The smaller instance of the dataset has 29 tasks that should be associated to 7 workstations. This represents a total of 10^{24} allocation possibilities, while the largest instance has a search space of the order of 10^{510}.

Table III shows the number of optimum solutions found by the GA, as well as the average relative and maximum deviation from optimum. Both the best results and the average are shown for a series of 15 runs of each of the 302 benchmark instances. A virtual machine with 2 GB RAM allocated memory and a 2.3 GHz processor was used.

The effect of the different heuristics can be seen in Table IV. The three procedures brings advantages compared to the simple GA. The exponential distribution focus the search on more prominent areas. This distribution gives to the

TABLE III. SUMMARY OF THE BEST AND AVERAGE RESULTS FOR THE 302 TESTED SALBP-2 INSTANCES

	# optimum	Relative deviation	Maximal deviation
Best allocation	127	0.79%	5.56%
Average	92	1.34%	5.94%

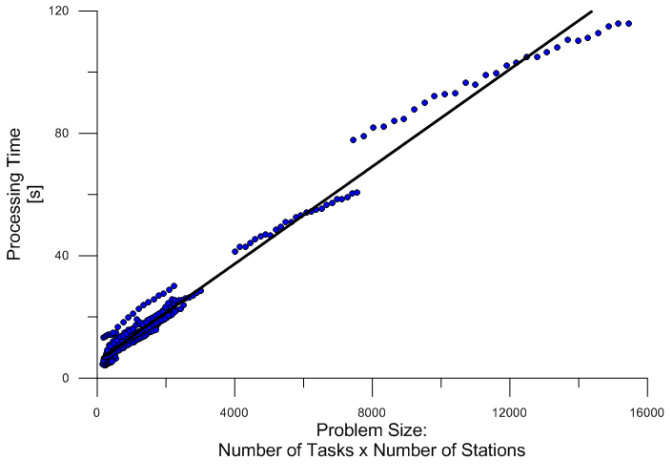

Fig. 5. Processing time versus problem size. Problem size is the number of station multiplied by the number of tasks.

GA some similarity with a greedy algorithm. The closing of workstations makes the assignments improve every time a new incumbent solution is found. It prevents tasks to be allocated in workstations that are already fully loaded. The lower and upper bound calculations uses the precedence graph information to set bounds for the tasks allocations. Once the cycle time is not known, the best answer is used as a time limit. Every time a new best solution is found, the bounds are updated. The sum of the effect of the heuristics results in an algorithm whose solutions are 0,79% far from optimum.

The average processing time was 25.0 seconds. Figure 5 shows the behavior of the processing time for the different SALBP problems. Once the same parameters for number of individuals and generations are used, the amount of time the algorithm uses is linearly dependent to the number of tasks and the number of allocation options which task has.

The SALOME algorithm [10] is used as a result comparison reference. Table V summarizes the compared results from the two methods. The results for SALOME are showed for a run time limit of 60 and 1000 seconds.

V. CONCLUSION

This paper presented a hybrid genetic algorithm to treat assignment restricted assembly line problems. The paper contributes with the use of heuristics in the encoding of the genetic algorithm. As indicated in Table IV, the GA worked efficiently

TABLE IV. SUMMARY OF THE RESULTS FOR THE EFFECT OF THE HEURISTICS.

	#optimum	Relative deviation	Maximal deviation
No heuristics	10	97.60%	479.08%
Exponential distribution	9	16.45%	75.34%
Workstation closing	71	9.87%	90.67%
Lower and upper bounds	78	9.57%	89.38%
Complete GA	127	0.79%	5.56%

TABLE V. METHOD COMPARISON FOR INSTANCES OF SALBP-2.

	#optimum	Rel. deviation	Max. deviation	Avg. time
GA	127	0.79%	5.56%	25.0 s
SALOME (60)	263	0.16%	5.88%	9.2 s
SALOME (1000)	273	0.13%	5.88%	123.3 s

only with the use of these heuristics. Without them, with the same conditions of 1000 generation and 200 individuals, the relative deviation from optimal was 97.6%. The variable reduction, the search focus in more promising regions and the heavy loaded workstation closing process helped to achieve answers with a relative deviation of 0,79% from the optimal solution. No convergence or stop criteria was used. So every instance was run for 1000 generations. Convergence controls or lower bounds knowledge could be used to decrease the processing time indicated in Table V.

The GA reached high quality answers on average; in 42% of the 302 tested instances, the optimum was found. Nevertheless, the GA is still underperforming when comparing to SALOME for SALBP. The set of heuristics used in SALOME showed to be superior to the evolution progress of the proposed GA.

The GA can be improved by using other strategies to define the task order allocation. For this work, the tasks were allocated by the numerical order. Most algorithms presented in Section II, however, uses 2 directions in the search procedure. Task renumbering procedures can also be used to force earlier assignments of larger tasks or the ones with more dependents.

The proposed GA can be used to solve both the TSALBP and the ARALBP cases. The restrictions needed in these variations can be implemented in the creation of the allocation possibility list of each task.

ACKNOWLEDGMENT

The authors would like to thanks the corrections and comments made by anonymous referees. We also would like to show our appreciation for the financial support from Fundação Araucária (Agreement 141/2015 FA-UTFPR-RENAULT) and CNPq (Grant 305405/2012-8).

REFERENCES

[1] M. E. Salveson, "The assembly line balancing problem," *The Journal of Industrial Engineering*, vol. 6, pp. 18–25, 1955.

[2] K. R. Baker, *Introduction to sequencing and scheduling*. New York: Wiley, 1974.

[3] N. Boysen, M. Fliedner, and A. Scholl, "Assembly line balancing: Which model to use when?" *International Journal of Production Economics*, vol. 111, no. 2, pp. 509–528, Feb. 2008.

[4] R. Gamberini, A. Grassi, and B. Rimini, "A new multi-objective heuristic algorithm for solving the stochastic assembly line re-balancing problem," *International Journal of Production Economics*, vol. 102, no. 2, pp. 226–243, Aug. 2006.

[5] E. Falkenauer, "Line Balancing in the Real World," in *Proceedings of the International Confe- rence on Product Lifecycle Management PLM 05*, Lyon, 2005, pp. 360–370.

[6] M. Rabbani, S. M. Kazemi, and N. Manavizadeh, "Mixed model U-line balancing type-1 problem: A new approach," *Journal of Manufacturing Systems*, vol. 31, pp. 131–138, 2012.

[7] G. Levetin, J. Rubinovitz, and B. Shnits, "A genetic algorithm for robotic assembly line balancing," *European Journal of Operational Research*, vol. 168, pp. 811–825, 1996.

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

[9] A. Scholl, M. Fliedner, and N. Boysen, "Absalom: Balancing assembly lines with assignment restrictions," *European Journal of Operational Research*, vol. 200, no. 3, pp. 688–701, Feb. 2010.

[10] R. Klein and A. Scholl, "Maximizing the production rate in simple assembly line balancing - A branch and bound procedure," *European Journal of Operational Research*, vol. 91, pp. 367–385, 1996.

[11] J. Sternatz, "Enhanced multi-Hoffmann heuristic for efficiently solving real-world assembly line balancing problems in automotive industry," *European Journal of Operational Research*, vol. 235, pp. 740–754, 2014.

[12] S. Akpnar and G. M. Bayhan, "A hybrid genetic algorithm for mixed model assembly line balancing problem with parallel workstations and zoning constraints," *Engineering Applications of Artificial Intelligence*, vol. 24, pp. 449–457, 2011.

[13] T. Pape, "Heuristics and lower bounds for the simple assembly line balancing problem type 1: Overview, computational tests and improvements," *European Journal of Operational Research*, vol. 240, pp. 32–42, 2015.

[14] Kim Y. K., Y. J. Kim, and Y. Kim, "Genetic algorithms for assembly line balancing with various objectives," *Computers & Industrial Engineering*, vol. 30, pp. 397–409, 1996.

[15] H. Gokcen and E. Erel, "Binary Integer Formulation for Mixed-Model Assembly Line Balancing Problem," *Computers & Industrial Engineering*, vol. 34, no. 2, pp. 451–461, 1998.

[16] A. Scholl and R. Klein, "Balancing assembly lines effectively A computational comparison," *European Journal of Operational Research*, vol. 114, no. 1, pp. 50–58, Apr. 1999.