Solving Assembly Line Balancing Problem using Genetic Algorithm with Heuristics-Treated Initial Population

[1]Kuan Eng Chong, Mohamed K. Omar, and Nooh Abu Bakar

***Abstract* — Although genetic algorithm (GA) has been widely used to address assembly line balancing problems (ALBP), not much attention has been given to the population initialization procedure. In this paper, a comparison is made between a randomly generated initial population and a heuristics-treated initial population. A heuristics-treated population is a mix of randomly and heuristics generated individuals in the initial population. Both populations are tested with a proposed GA using established test problems from literature. The GA, using a fitness function based on realized cycle time is capable of generating good solutions.**

***Index Terms* — assembly line balancing, genetic algorithms, manufacturing optimization, realized cycle time.**

I. Introduction

Installing an assembly line is a long-term and a costly decision. It is therefore imperative that the assembly line must be well designed and properly balanced to ensure maximum efficiency.

The classical decision problem of optimally balancing the assembly line is known as simple assembly line balancing problem (SALBP). SALBP can be classified by its objective function where the problem versions include the SALBP -1, SALBP-2, SALBP-F and SALBP-E [1], [2]. The objective of the SALBP-1 problem is to minimize the number of workstations for a given cycle time, whereas SALBP-2 problem minimizes the cycle time given a predetermined number of workstations. Unlike the previous two versions, SALBP-F determines whether or not a feasible assembly configuration exists for a given combination of cycle time and number of workstations. Lastly, the SALBP -E attempts to maximize the line efficiency by minimizing the number of workstations and cycle time simultaneously.

The SALBP is a combinatorial optimization problem and extensive research has been conducted to solve this problem and its variants. Exact methods proposed are able to generate optimum solution(s). Some exact procedures found in literature include the application of dynamic and integer programming formulations and branch and bound procedures [1]. Since SALBP is classified as NP-hard, solving it optimally by total enumeration is not practical with real-world or large-sized problems. Researchers shift their focus towards heuristics approaches as a popular way to address hard problems. Heuristics are efficient as they are fast and simple to implement. Heuristics cannot guarantee optimality, but are able to seek good solutions at a reasonable computational cost [3]. Maximum numbers of immediate followers, [4], largest candidate rule [5], COMSOAL, [6] Hoffman's enumeration method, [7], MALB [8], are some of the many examples of heuristics reported in literature. Please refer to [9]-[12] for a more comprehensive reviews and comparative evaluations on heuristic solutions for SALBP.

As most heuristics are generally problem specific, their applications are limited. The focus of research thus shifts towards the development of powerful metaheuristic algorithms. A metaheuristic provides a general algorithmic framework which can be applied to various optimization problems. Classification of metaheuristics includes simulated annealing, tabu search, iterated local search and evolutionary computing. Evolutionary computing in general refers to several heuristic techniques based on the principles of natural evolution. One of these heuristics is genetic algorithms.

II. Genetic Algorithms

Genetic algorithms are stochastic search techniques that mimic the natural process of evolution. Since its introduction in the 1970's [13], GA has gained wide acceptance in many different fields of research. GA has been proven to be a powerful tool to find approximate solutions to large combinatorial optimization problems. In recent years the use of GA to solve assembly line balancing problems has also been increasing in numbers. For more detailed review on the application of GA in assembly line balancing, please refer to [14].

The main sequence of the GA procedure is as simplified below:

Step 1: Generate initial population
Step 2: Evaluate fitness of each individual in the population
Step 3: Select individuals for reproduction

[1]Kuan Eng Chong is with the Faculty of Manufacturing Engineering, Technical University of Malaysia, 75450 Malacca, Malaysia (phone: +60126081966; fax: +6062332414; e-mail: kuaneng@utem.edu.my).

Mohamed K. Omar is with the Faculty of Engineering and Technology, Multimedia University, 75450 Malacca, Malaysia (email: mohamed.k.omar@mmu.edu.my).

Nooh Abu Bakar is with the Business & Advanced Technology Centre, University Technology Malaysia-City Campus, 54100 Selangor, Malaysia (email: noohab@citycampus.utm.my)

Step 4: Apply genetic operators to create the next generation
Step 5: If terminate criteria are met, end GA. If the criteria are not met, go to Step 3

In this paper, a GA procedure is proposed to solve SALBP-1. The following section describes the features and parameters of the proposed GA.

A. Chromosome Representation and Initial Population

The proposed GA adopts the task-based encoding scheme where a chromosome is represented by a feasible sequence of tasks. The length of the chromosome is determined by the number of tasks in the precedence diagram. A typical population size of 100 chromosomes or individuals makes up the initial population [14]. A procedure similar to COMSOAL [6] is employed to generate random individuals in the initial population. Each individual consist of a feasible sequence of tasks. A greedy heuristic using the station oriented approach is then applied to assign these tasks to workstations. Task assignment by station oriented approach is reported to perform better than task oriented approach [15], [16]. Tasks are loaded into a workstation as long as the total workstation load does not exceed the prescribed cycle time. If cycle time is exceeded a new workstation will be opened and the assignment process continues as before. The above procedure is used to generate the random initial population.

A heuristics-treated initial population is next generated. A heuristics-treated population is a mix of randomly and heuristic generated individuals in the initial population. Two new individuals are generated by two different heuristics and introduced into the initial population. The heuristics employed are the ranked positional weight (RPW) technique and the largest candidate rule. The RPW technique was introduced by Helgeson and Bernie [17], where work elements are ranked according to their respective positional weight. The largest candidate rule [5] assigns tasks to workstations in the similar manner as the RPW approach but tasks are instead ranked based on the task durations.

The randomly generated initial population and the heuristics-treated initial populations are then tested with various test problems (see section IV) and their results are discussed in section V.

B. Fitness Function

Most of the literature on GA applications in solving assembly line balancing problems focused on SALBP-1, [14]. These GA procedures prefer to use the balance delay and its variants as fitness functions. Balance delay is defined as:

$$Balance\ delay = \sum_{j=1}^{m} (c - S_j)$$

where

m is the total number of workstations
c is the predetermined cycle time
S_j is the time of workstation j, j=1…m

Driscoll and Thilakawardana [18] introduced a new performance measure, the line efficiency, which has several benefits over the balance delay. The line efficiency is dimensionless and scaled in percentage form, making results more meaningful and easily interpreted. The new measure is hence representative of line utilization [18].

We modified this performance measure by using realized cycle time instead of the predetermined cycle time and incorporate this as our fitness function. The objective is to maximize the line efficiency. The fitness function is shown below:

$$line\ efficiency = \frac{\sum_{i=1}^{n}(t_i)}{m \times c_r} \times 100$$

where

n is total number of tasks
t_i is duration of task i, i=1,…n
m is total number of workstations
c_r is the realized cycle time

Realized cycle time is defined as the maximal station time after the tasks assignment process [2]. Realized cycle time can be equal to or smaller than the prescribed cycle time. Although the primary objective of SALBP-1 is the minimization of workstations, the use of realized cycle time is more accurate fitness indicator for solutions with the same number of workstation(s). Fig. 1 below illustrates a precedence diagram with 6 tasks. Processing time is given next to each task. To solve SALBP-1 for cycle time of 10 time units, the solution at the top yields 3 workstations and realized cycle time of 10 time units (from workstation 3) with line efficiency of 86.67% . The solution at bottom also yields 3 workstations but with realized cycle time of 9 time units (from workstations 1 & 2) and 96.3% line efficiency. Both solutions are feasible but the second configuration is more superior in terms of better line efficiency.

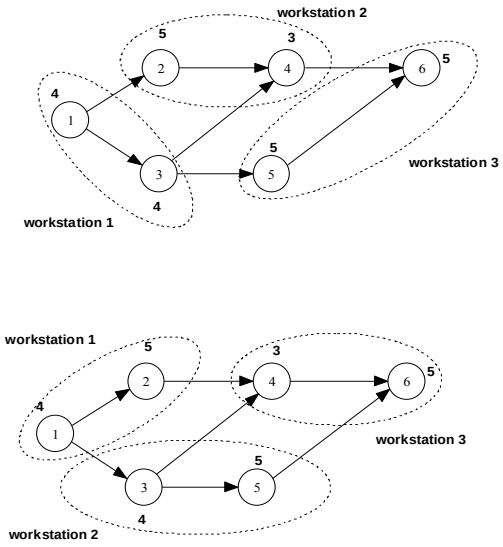

Fig. 1. Solutions for SALBP-1 with cycle time=10 time units

C. Selection and Reproduction

The selection of individuals for reproduction is performed by the roulette wheel method. This popular approach allows individuals with greater fitness, a better chance of being selected for reproducing the next generation [19], [20].
The proposed GA also uses the "elitism" strategy to improve the GA performance by retaining the best individual in each generation. This is to ensure that good individuals are not lost or destroyed by the crossover and mutation operators [21], [22].

Standard crossover and mutation operators will generate infeasible individuals in assembly line balancing problems because of the precedence constraints. Researches have developed specialized crossover and mutation operators for ALBP which ensure chromosome feasibility. Our GA uses a modified version of the 2-point order crossover operation and the scrambled mutation operation proposed by Leu, Matheson and Rees [23]

D. Stopping Criteria

The GA will terminate when either one of the two following stopping criteria is met: maximum generations or stalled generations. Maximum generations is maximum number of iterations the GA will perform (set to 500). Stall generations is the number of iterations with no improvement in the best fitness value (set to 300).

III. Problem Statement

Solving actual industry assembly line balancing problems is difficult with the many real world constraints. To tackle more complex problems, GA must be able to produce good solutions at the shortest possible time. There are many parameters that can influence the performance of a GA. Tasan and Tunali [14] in their review on GA applications in assembly line balancing problems noted that many researches focused on efficient encoding schemes to handle difficult constraints and infeasibility. However, there is little attention given to the influence of population initialization process in solving SALBP. So far, Leu, Matheson and Rees [23], suggest that heuristic generated solution should be included in the initial population to produce better solutions.

The paper hopes to address this issue by comparing the performance of the proposed GA with randomly generated initial population and heuristics-treated initial population.

IV. Solution Methodology And Computational Experience

The proposed GA is coded in Matlab 7.0 and executed on a Windows-based, Intel Pentium M, 1.50GHz computer. Matlab is chosen as the development platform because the availability of the Genetic Algorithm toolbox and useful built-in functions. However, several of the standard genetic operators and procedures available in the toolbox need to be modified to meet our problem specifications.

The proposed GA is tested with established SALBP-1 test problems from literature so that we can make a comparative evaluation between published results and the proposed GA's results. We use the Hoffman's dataset [24] as it is provides a wide range of established problems. From this dataset, seven test problems were selected, based on task sizes and complexity. The smallest test problem is the Bowman problem with 8 tasks and the largest is the Arcus problem with 111 tasks. Each test problem is constructed using data from [25] and solved for different cycle times, identical to those cycle times in the Hoffman's dataset.

The performances of both initial populations are assessed by 2 criteria: (i) the number of workstations and (ii) realized cycle time. For each cycle time, the GA is first tested with the randomly generated initial population and the number of workstations generated and realized cycle times are recorded. The same process is repeated for heuristics-treated initial population.

V. Results And Discussions

Table I shows the results produced by the proposed GA for the two initial populations. The selected test problems and their size (number of tasks) are shown in the first and second columns respectively. This is followed by the cycle times for each problem in the third column. The next two columns display the number of workstations and realized cycle time of the solutions of GA using randomly generated initial population. The last two columns are the results from GA with heuristic-treated initial population.

An initial population is considered to perform better if the GA produces solutions with lesser number of workstations. If both solutions produce the same number of workstations, the performance is determined by the realized cycle time.

The results show that for smaller sized problems (Bowman, Jackson, Mitchell and Sawyer test problems), there is not much difference in performance for both populations. The reason is, possibly, for problems with a few tasks, the search space is small and the GA is able to quickly locate the best solution, without or without the heuristics solutions.

The benefit of using heuristics-treated initial population is only apparent in large-sized test problems (Killbridge, Tonge, and both Arcus problems) where the search space is much larger. Heuristics-treated population performed better by either producing configurations with lower numbers of workstations or better realized cycle time. For example, the Tonge test problem with cycle time of 176 time units, the GA with heuristics-treated population produces lower number of workstations (21 workstations) compared to the GA with random population (22 workstations). From the same test problem but with cycle time of 410 time units, GA with both initial populations generates the same number of workstations (9 workstations) but the heuristics-treated population yields better realized cycle time of 397 time units compared with 398 time units for the randomly generated initial population.

From a total of 44 tests, the heuristics-treated initial population outperforms the randomly generated population in 14 instances (31.8 %) and performs equally well in 30 instances (68.2 %), as shown in Table II.

It should also be noted that the performance improvements of heuristics-treated population are concentrated in large-sized test problems. For example, for the Arcus problem with 111 tasks, heuristics –treated population performs better in 5 out of 6 instances whereas in the Mitchell problem, only 1 out of 6 instances provides better solution. Hence, if we just focus our test on bigger size problems, the percentage improvement will be much larger.

In this preliminary study, the results concur with previous findings [23] that heuristics-treated initial population can perform better than randomly generated initial population. Studies using GA for SALBP are therefore encouraged to use heuristic generated individuals in their initial population.

TABLE II
RESULTS TO COMPARE PERFORMANCE OF GA WITH HEURISTICS –TREATED INITIAL POPULATION OVER RANDOMLY GENERATED INITIAL POPULATION

	Performs better	Same	Performs worst
Instances	14	30	0
Percentage	31.8 %	68.2%	0%

VI. CONCLUSIONS

Although the primary objective of SALBP-1 is to minimize the number of workstations given a predetermined cycle time, GA can generate alternative solutions with different realized cycle time. In this study, we presented a GA procedure using line efficiency based on realized cycle time as the fitness function. This GA is capable of producing good results when solving SALBP-1 using test problems from literature. We also compared randomly generated initial population with heuristics-treated initial population. Test results indicate that heuristics-treated initial population performs better for large-sized problems. This is encouraging as in the real world most assembly line problems are complex. We suggest more research on developing better initial population to improve the performance of GAs used in SALBP.

This research is motivated by the assembly line design problems confronted by a consumer electronics manufacturer. High demands and short product shelf life compels the company to operate at short cycle times and at tight schedules. With the current scenario, any small reduction in cycle time can translate into big improvements to production capacity and cost savings. Also, with rapidly changing product specifications, the company is constantly creating new assembly lines for the production of new models. The contribution from this study will provide new knowledge to develop a better GA tool to solve real world problems more efficiently at the shortest possible time.

REFERENCES

[1] I. Baybars, "A survey of exact algorithms for the simple assembly line balancing problem,"*Management Science*, 32, 1986, pp. 909-932.

[2] A. Scholl, "*Balancing and sequencing of assembly lines*," Heidelberg:Physica-Verlag, 1999, ch. 2.

[3] C.R. Reeves, "*Modern heuristic techniques fro combinatorial problems*," New York: John Wiley & Sons, 1993, ch. 1.

[4] F. M. Tonge, "A heuristic program of assembly line balancing," Prentice-Hall Englewood Cliffs, NJ , 1961.

[5] C. L. Moodie, and H. H. Young, "A heuristic method of assembly line balancing for assumptions of constant or variable work element times," *Journal of Industrial Engineering,* vol. 16, 1965, pp. 23-29.

[6] A. L. Arcus, "COMSOAL: A computer method of sequencing operations for assembly lines," *International Journal of Production Research*, vol. 4, 1966, pp. 259-277

[7] T. R. Hoffman, "Assembly line balancing with precedence matrix," Management Science, vol. 9, 1963, pp. 551-562.

[8] E. M. Dar-El, "MALB-A heuristic technique for balancing large single-model assembly lines," *AIIE Transactions*, vol. 5, 1973, pp. 343-356.

[9] F. B. Talbot, J. H. Pattrson, and W. V. Gehrlein, "A comparative evaluation of heuristic line balancing techniques," *Management Science*, vol. 30, 1986, pp.430-454.

[10] A. Scholl and S. Voβ, "Simple assembly line balancing-heuristic approaches," *Journal of Heuristics*, vol. 2, 1996, pp. 217-244.

[11] E. Erel and S. C. Sarin, "A survey of assembly line balancing procedures," *Production, Planning and Control*, vol. 9, 1998, pp. 414-434.

[12] A. Scholl and C. Becker, "State-of-the-art exact and heuristic solution procedures for simple assembly line balancing," *European Journal of Operation Research*, vol. 168, 2006, pp. 666-693.

[13] J. H. Holland, "Adaptation in natural and artificial system," Ann Arbor, Michigan:The University of Michigan Press, 1975.

[14] S. O. Tasan and S. Tunali, "A review of the current applications of genetic algorithms in assembly line balancing," *Journal of Intelligent Manufacturing*, vol. 19, 2008, pp.49-69.

[15] A. Scholl and S. Voβ, "*A note on fast, effective heuristics for simple assembly line balancing*," (working paper), TH Darmstadt, 1994.

[16] J. F. Gonçalves and J. R. De Almeida, "A hybrid genetic algorithm for assembly line balancing," ," *Journal of Heuristics*, vol. 8, 2002, pp. 629-642.

[17] W. B. Helgeson and D. P. Birnie, "Assembly line balancing using ranked positional weight technique," *Journal of Industrial Engineering* , vol. 12, 1961, pp. 394-398.

[18] J. Driscoll and D. Thilakawardana, "The definition of assembly line balancing difficulty and evaluation of balance solution quality," *Robotics and Computer Integrated Manufacturing* , vol. 17, 2001, pp. 81-86.

[19] D. A. Coley, "*An introduction to genetic algorithms for scientists and engineers*," Singapore: World Scientific Publishing Co., 2001, pp.23-25.

[20] Z. Michalewicz, "*Genetic algorithms+data structures=evolution programs*," Springer-Verlag Berlin Heidelberg, 1996, ch. 2.

[21] D. E. Goldberg, "*Genetic algorithms in search, optimization, and machine learning*," Addison-Wesley Publishing Company, Inc. 1989.

[22] M. Mitchell, "*An introduction to genetic algorithm*," 1996, The MIT Press, pp.168

[23] Y. Y. Leu, L. A. Matheson and L. P. Rees, "Assembly line balancing using genetic algorithm with heuristic generated initial population and multiple criteria," *Decision Science,* vol. 15, pp. 581-606.

[24] T. R. Hoffman, "Assembly line balancing: a set of challenging problems," International *Journal of Production Research*, vol. 28, 1990, 1807-1815.

[25] A. Scholl, "*Data of assembly line balancing problems*," (working paper), TH Darmstadt, 1993.
Available: http://www.assembly-line-balancing.de/

TABLE I
RESULTS COMPARING RANDOM AND HEURISTIC-TREATED INITIAL POPULATION FOR TEST PROBLEMS FROM LITERATURE

Test problem	No. of tasks	Cycle time	Randomly generated population		Heuristics-treated population	
			workstations	realized cycle time	workstations	realized cycle time
Bowman	8	20	5	17	5	17
Jackson	11	7	8	7	8	7
		9	6	9	6	9
		10	5	10	5	10
		13	4	12	4	12
		14	4	12	4	12
		21	3	16	3	16
Mitchell	21	14	9	13	8	14
		15	8	15	8	15
		21	5	21	5	21
		26	5	23	5	23
		35	3	35	3	35
		39	3	36	3	36
Sawyer	30	25	14	25	14	25
		27	13	26	13	26
		30	12	30	12	30
		36	10	35	10	35
		41	9	40	8	41
		54	7	48	7	48
		75	5	66	5	66
Kilbridge	45	57	10	57	10	57
		79	8	71	7	79
		92	7	81	7	81
		110	6	94	6	94
		138	5	117	4	138
		184	3	184	3	184
Tonge	70	176	22	175	21	176
		364	10	357	10	357
		410	9	398	9	397
		468	8	446	8	446
		527	7	506	7	506
Arcus	83	5048	16	4943	16	4943
		5833	14	5724	14	5621
		6842	12	6659	12	6591
		7571	11	7141	11	7141
		8412	10	8036	10	7882
		8898	9	8528	9	8528
		10816	8	10306	8	10306
Arcus	111	5755	28	5689	27	5752
		8847	19	8265	18	8689
		10027	16	9736	16	9684
		10743	15	10323	15	10288
		11378	14	11121	14	11121
		17067	9	16885	9	16872

ISBN:978-988-17012-3-7