RDU140103

MODELLING AND OPTIMIZATION OF ASSEMBLY LINE BALANCING WITH MULTI-RESOURCE CONSTRAINTS

MOHD FADZIL FAISAE AB. RASHID
RAZMAN MAT TAHAR
NIK MOHD ZUKI NIK MOHAMED
AHMAD NASSER MOHD ROSE
MOHAMAD ZAIRI BAHAROM

RESEARCH VOTE NO:
RDU140103

Faculty of Mechanical Engineering
Universiti Malaysia Pahang

2017

Special dedication to our co-researcher, Allahyarham Professor Dr. Razman Mat Tahar.

ACKNOWLEDGEMENTS

Alhamdulillah, through His blessing, this research was finally successfully conducted. First and foremost, we would like to acknowledge Ministry of Higher Education, Malaysia for funding this research. We also would like to thanks Universiti Malaysia Pahang and Faculty of Mechanical Engineering for providing necessary supports.

This acknowledgement is also dedicated to all co-researchers, academic, administration and technical staff in Faculty of Mechanical Engineering and Faculty of Manufacturing Engineering, Universiti Malaysia Pahang for all the ideas, support and cooperation to make this research succeed. Finally to all the students who were benefits from this research, make this research as your beginning step towards excellence.

ABSTRACT

Assembly Line Balancing (ALB) is an attempt to assign tasks to various workstations along a line so that the precedence relations are satisfied and performance measures are optimised. Assumption by previous researches that any assembly task can be performed in any workstation encourages the author to focus on the resource usage in ALB. Limited number of resources in the industry also becomes a vital influencer to consider this constraint in ALB. This research aim to model and optimize the ALB with resource constraints. In different with existing work that assume all workstations have similar capability, this research consider the tool, worker and equipment constraints in the assembly process. The early study reveal that the current trend in production line is becoming more complex due to growth in the product development and also market force to minimize the set up cost. Therefore, beside only consider the simple assembly line problem (SALB), this research also consider two-sided assembly line problem (2S-ALB) and mixed-model assembly line problem (MMALB) that match with current trend in assembly layout. All three version of ALB problems have been modelled to include the general resource constraints. This make all three versions of ALB model is unique compared with existing model in literature. In all three ALB version, the resource constraints have been set as one of objective function to be minimized. The optimization was conducted by using different optimization algorithms. In SALB, we found that the Elitist Non-Dominated Sorting Genetic Algorithm (NSGA-II) have better performance. Meanwhile, for 2S-ALB and MMALB, the Particle Swarm Optimization and Ant colony Optimization have superior performance respectively. The validation for the problem ALB with resource constraints have been conducted in electronic and automotive assembly plant. The final result indicated that the proposed model is capable to improve the assembly efficiency and also capable to reduce the number of the resources required in assembly plant.

ABSTRAK

Pengimbangan rangkaian pemasangan (ALB) ialah proses mengagihkan kerja pemasangan ke stesyen-stesyen kerja supaya kekangan turutan dapat dipenuhi dan prestasi pemasangan dapat dioptimakan. Andaian daripada penyelidik terdahulu, di mana menyatakan mana-mana kerja boleh dijalankan pada mana-mana stesyen kerja adalah tidak benar. Keterhadan bilangan sumber di industri juga menjadi sebab keperluan kajian ini. Kajian ini mensasarkan untuk memodel dan mengoptimakan ALB dengan kekangan sumber. Berbeza dengan kajian terdahulu yang menganggap semua stesyen kerja mempunyai keupayaan yang sama, kajian ini mengambilkira kekangan dari segi peralatan, pekerja dan mesin di dalam proses pemasangan. Kajian awal mendedahkan trend semasa di rangkaian pemasangan yang menjadi semakin kompleks di sebabkan pembangunan produk dan tekanan pasaran untuk meminimakan kos. Oleh itu, selain dari masalah rangkaian pemasangan mudah (SALB), kajian ini juga mengambilkira rangkaian pemasangan dua arah (2S-ALB) dan rangkaian pemasanagan produk bercampur (MMALB). Semua tiga versi ALB ini telah dimodelkan untuk mengambilkira kekangan sumber secara umum. Ini menjadikan ketiga-tiga model ini unik berbanding dengan kajian sebelumnya. Di dalam ketiga-tiga variasi ALB, kekangan sumber dijadikan sebagai salah satu objektif pengoptimaan. Pengoptimaan telah dilakukan menggunakan algoritma yang berbeza. Bagi SALB, didapati Elitist Non-Dominated Sorting Genetic Algorithm (NSGA-II) mempunyai prestasi yang lebih baik. Manakala bagi 2S-ALB dan MMALB, Particle Swarm Optimization dan Ant colony Optimization masing-masing mempunyai kelebihan berbanding algoritma lain. Proses validasi untuk ALB dengan kekangan sumber ini dilakukan di industri pemasangan elektronik dan automotif. Hasil daripada validasi ini menunjukkan model yang dicadangkan berupaya untuk meningkatkan kecekapan pemasangan dan mengurangkan jumlah sumber di dalam proses pemasangan.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

This chapter gives a brief description of the research background including the problem statement, followed by research objectives, research scope and significant of research. In the next section, the structure of this thesis is briefly explained and in the last part, the chapter summary is presented.

1.2 RESEARCH BACKGROUND

Assembly line is one of the industrial production systems used to produce finished goods in an industry. It has been widely employed in many production industries such as automotive, electronics and other consumer durable production to enhance the efficiency of production system (Mozdgir et al., 2013). The rapid development of manufacturing industry is caused by increasing customer demands. This scenario forced manufacturers to maximise the production output in order to meet customers' demands. This can be achieved by eliminating process inefficiencies (i.e. minimise the number of workstations and cycle time) as well as by utilising resources at an optimum level.

Assembly Line Balancing (ALB) problem is defined as assigning tasks to workstations to optimise some performance measures by reducing the percentage of idle time or balance delay of assembly line (Chen et al., 2006; Ranjan and Pawar, 2014). This research aims to maximise the production rate and achieve the number of workstations, while satisfying some particular constraints, such as (i) precedence constraints and (ii) the total processing time assigned to each workstation must not exceed the cycle time (Suwannarongsri and Puangdownreong, 2008).

Due to limited number of resources in industry, it is necessary to consider the problem in assembly optimisation. This research intends to focus on multi-objective optimisation of Assembly Line Balancing Type-E (ALB-E) problem of a simple model.

1.3 PROBLEM STATEMENT

The main problem with the existing research in the ALB is the assumption that any assembly task can be performed at any workstation (Scholl & Becker, 2006; Zhang et al., 2007; Zhang et al., 2008; Hamta et al., 2013). However, each workstation has its own capabilities and specialisation. This situation has been highlighted as one of the serious problems in the industry (Ağpak and Gökçen, 2005; Bautista and Pereira, 2007). This finding is consistent with the study by Sungur and Yavuz, (2015) that emphasizes workers' assignment to be mandatorily based on their qualification.

Rapid growth in manufacturing becomes a vital influencer to consider the usage of resources due to limited number of machines and tools. Although a small number of researches considered resource constraint in their works, none of them focused on resource constraint in ALB especially in terms of machine, tool and worker constraints (Ağpak and Gökçen, 2005; Browning and Yassine, 2010; Corominas et al., 2011 Battaïa and Dolgui, 2013). It is important to consider these constraints due to the limited number of resources where the utilisation of these resources can be minimised.

In the past years, the Genetic Algorithm (GA) approach has attracted the attention of researchers to solve issues related to ALB (Gurevsky at al., 2013; Zacharia and Nearchou, 2013; Al-Hawari et al., 2014). This finding is consistent with the finding of past studies that used similar approach (Scholl & Becker, 2006; Suwannarongsri & Puangdownreong, 2008; Wei & Chao, 2011). Till date, to the best knowledge of the researcher, none of the published work employed Elitist Non-Dominated Sorting Genetic Algorithm (NSGA-II) in the optimisation of ALB-E in terms of resource constraint.

For that reason, it is crucial to propose NSGA-II to address the research gap as it is capable to solve real-world optimisation problem for multi-objective functions (Chica et al., 2012; Guo et al., 2014; Zhao et al., 2015). In comparison with the old version of Non-Dominated Sorting Genetic Algorithm (NSGA), the NSGA-II implements elitism-preserving technique (Deb, 2001). The findings from past studies by Deb et al. (2002) and Zhao et al. (2015) concluded that NSGA-II has better convergence towards Pareto-optimal front. The solutions generated by NSGA-II are geared towards Pareto-optimal front. However, the existing algorithms such as Multi-Objective Genetic Algorithm (MOGA) usually have slow convergence (Fonseca and Fleming, 1993). Based on literature review, there are no studies available on the implementation of NSGA-II to optimise the ALB-E with resource constraint and this has motivated the researchers to conduct the present study.

1.4 RESEARCH OBJECTIVE

The objectives of this research are:

i. To study the assembly line balancing (ALB) problem and establish a mathematical model for ALB with resource constraints.
ii. To propose and optimise suitable optimization algorithm for the ALB with resource constraints.
iii. To validate the mathematical model and optimisation of algorithm through an industrial case study.

1.5 RESEARCH SCOPE

The scope of this research are stated as follow:

i. This research studies on the optimisation of Assembly Line Balancing with resource constraint. The scope of problem is limited to three version of ALB problems:
    a. A simple ALB type E problem (ALB-E) with resource constraints.
    b. Two-sided ALB (2S-ALB) with resource constraints.
    c. Mixed-model ALB (MMALB) with resource constraints.

ii. In this research, different optimization algorithms are tested for different ALB problem types. However, the algorithms are limited to three main metaheuristics; Genetic Algorithm (GA), Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO).

iii. The validation method of the optimised algorithm is conducted using test problems from literature. In addition to that, the algorithm and mathematical model are validated through an industrial case study to ensure the applicability of the optimisation results. The industrial case study is only focused on and conducted in an electronic company.

1.6 SIGNIFICANCE OF RESEARCH

By achieving the aforementioned objectives, the research will increase the efficiency of assembly process. This research targets to reduce the number of workstation in an assembly as well as to minimise the cycle time by managing the number of resource used. Lower cycle time and number of workstation used will enhance the line efficiency. Apart from that, by considering the resource constraints, the resource usage will be significantly reduced in an assembly process.

By proposing an efficient way using the proposed algorithm and mathematical model to assemble a product, the long term implication of this research will be reflected on the enhanced industrial productivity. The modelling phase involves the steps to transform a product into a precedence diagram and also the steps on how to transform the precedence diagram into a digital format language that can be understood by a computer. Then, the NSGA-II will find the optimal solutions according to the objective functions to assemble a product.

CHAPTER 2

LITERATURE REVIEW

This chapter serves as a platform to review literature. The literature for this topic is presented in two papers as follow:

1. Jusop, M. and Ab. Rashid, M.F.F. (2015), "A Review On Assembly Line Balancing Type-E Problem", IOP Conf. Series: Materials Science and Engineering, vol. 100, 012005.
2. Abdullah Make, M.R., Ab. Rashid, M.F.F. and Razali, M.M. (2017), "A Review of Two-Sided Assembly Line Balancing Problem", The International Journal of Advanced Manufacturing Technology, vol. 89, Issue 5, pp. 1743-1763 (ISI IF=1.568).

A review on simple assembly line balancing type-e problem

A review on simple assembly line balancing type-e problem

M Jusop[1] and M F F Ab Rashid[1]
[1]Manufacturing Focus Group, Faculty of Mechanical Engineering, Universiti Malaysia Pahang, 26600 Pekan, Pahang, Malaysia

E-mail: masitahjusop@yahoo.com

Abstract. Simple assembly line balancing (SALB) is an attempt to assign the tasks to the various workstations along the line so that the precedence relations are satisfied and some performance measure are optimised. Advanced approach of algorithm is necessary to solve large-scale problems as SALB is a class of NP-hard. Only a few studies are focusing on simple assembly line balancing of Type-E problem (SALB-E) since it is a general and complex problem. SALB-E problem is one of SALB problem which consider the number of workstation and the cycle time simultaneously for the purpose of maximising the line efficiency. This paper review previous works that has been done in order to optimise SALB-E problem. Besides that, this paper also reviewed the Genetic Algorithm approach that has been used to optimise SALB-E. From the reviewed that has been done, it was found that none of the existing works are concern on the resource constraint in the SALB-E problem especially on machine and tool constraints. The research on SALB-E will contribute to the improvement of productivity in real industrial application.

1. Introduction

An assembly line is a manufacturing process comprises of a sequence of workstations in which a set of necessary task to assemble a product are performed. The development of assembly is system usually used in the production of goods in the industry. The idle time and the number of workstations on the production line have to be minimised whereas the line efficiency has to be maximised so as to achieve a balance line.

The decision problem of optimally partitioning the assembly task among the workstations with respect to some objective is known as Simple Assembly Line Balancing (SALB) [1]. This problem intends at grouping assembly operations which have to be performed to produce final products, and assigning the groups of operations to workstations, so as to make sure the total assembly time required at each station is nearly the same and the precedence constraints between operations are respected [2]. SALB is a type of NP-hard optimisation problems which means that when the number of assembly task is increased, the feasible solution will rise staggeringly [3-5]. Advanced approach of algorithm is necessary to solve large-scale problems.

SALB can be classified into two categories (i) Simple Assembly Line Balancing Problems (SALBP) (ii) General Assembly Line Balancing Problems (GALBP) [6, 7]. The most well-known assembly line is called simple assembly line balancing problem. Simple assembly line balancing is considered when the same product is running on the line. This type of problem is classified into four groups with respect to the objectives function [6, 8].

- Simple assembly line balancing Type-1 (SALB-1) aims to minimise the number of workstations on the line for a fixed cycle time.
- Simple assembly line balancing Type-2 (SALB-2) aims to minimise the cycle time for fixed number of workstations on the line.
- Simple assembly line balancing Type-E (SALB-E) aims to maximise the efficiency of the line simultaneously minimising the number of workstations and the cycle time.
- Simple assembly line balancing Type-F (SALB-F) aims to determine a feasible line for a combination of the number of workstations and cycle time.

Other problems which are not included in simple assembly line are considered as generalised assembly line balancing problems. Mixed-model assembly line balancing (MALBP) or mixed-model sequencing problem (MSP) and also U-line balancing problem (UALBP) are categorised as GALBP [7]. The classification of assembly line balancing problems is illustrated as in figure 1.

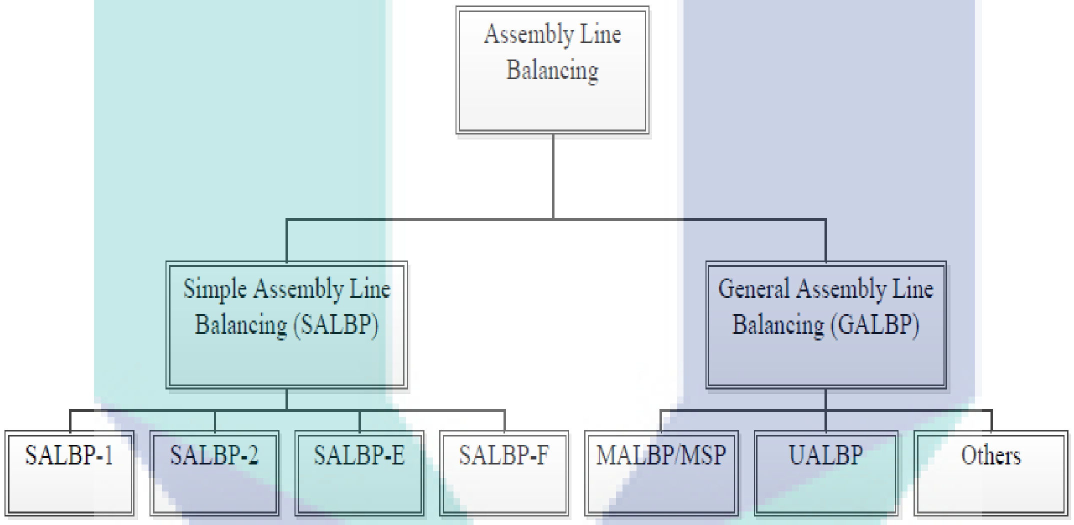

Assembly Line Balancing
Simple Assembly Line Balancing (SALBP)
General Assembly Line Balancing (GALBP)
SALBP-1
SALBP-2
SALBP-E
SALBP-F
MALBP/MSP
UALBP
Others

Figure 1. Classification of assembly line balancing problems.

Most of previous researches are focusing on SALB-1 [5, 6, 9, 10] and SALB-2 [2, 11-13]. Only a small number of previous research study on SALB-E as it is more complicated compare with SALB-1 and SALB-2. Study on SALB-E need to consider multi-objective functions instead of single objective in both SALB-1 and SALB-2. In real manufacturing scenario, it is better if we consider both parameters; minimised the number of workstations and minimised the cycle time for the purpose to maximise the assembly efficiency.

This paper reviews the previous study on simple assembly line balancing Type-E. The rest of the paper consists of problem modelling and objective function, SALB-E optimisation algorithm, and genetic algorithm for SALB-E. Finally, conclusion and suggestion for future research are addressed.

2. Problem modelling and objective function

Simple Assembly Line Balancing of Type-E Problem (SALBP-E) has been reviewed by Gurevsky et al. under dissimilarities of task processing times [14]. The research on stability of feasible and optimal solutions for SALBP-E is presented in this paper. Two heuristic procedures are proposed and evaluated on certain targets in order to find a concession between the two goal functions. Polynomial time algorithm has been proposed so as to compute the stability radius of feasible balances.

The paper presented by Suwannarongsri & Puangdownreong proposed a combination of partial random permutation (PRP) method and an adaptive tabu search (ATS) in an attempt to specify the optimum solutions for the assembly line balancing problem [15]. The researcher has considered the simple assembly line balancing in the work with four objective functions (i) minimise the number of workstations, (ii) minimise the idle time, (iii) minimise the workload variance and (iv)maximise the line efficiency. The equation (1) is used to represent the line efficiency.

$$E = \sum_{i=1}^{m} T_i / (mc) \tag{1}$$

where E : Line efficiency
m: Number of workstations
c : Cycle time
T_i: processing time of the i^{th} workstation

A test against three benchmark single-model SALB problems such as Buxey, Sawyer, and Warnecke on actual SALB problem has been conducted by the researcher to assure the efficiency of the proposed multiple-objective method. The results shows that the proposed method is efficient for multiple-objective compare to the single-objective.

Previous study by Scholl & Becker stated that there is no direct method to solve the SALBP-E [6]. That type of model can be solved by a search method; the combination of the number of stations m and the cycle time c which is feasible for the efficient line is chosen among the others or, the value of required line capacity as in equation (2) should be minimal.

$$T = m.c \tag{2}$$

where T is line capacity

The review published by Wei & Chao are focused on SALBP-E in order to optimise the line balancing efficiency as well as minimising the idle time [16]. This objective can be achieved by minimising the number of stations and the cycle time. SALBP-1 and SALBP-2 models are combined by the researcher in order to develop the SALBP-E model. In SALBP-1, the number of stations is minimised with fixed cycle time. This model is re-defined to SALBP-1-i with the intention of determining the minimum number of stations. The goal of modified model SALBP-2 is to ensure the minimisation of cycle time ct with a fixed number of workstations m. The efficiency of the line is formulated as equation (3):

$$E = \frac{t_{sum}}{m.ct} \tag{3}$$

where t_{sum} is the total time of all tasks

In order to maximise the line efficiency, the optimal number of workstation must be obtained by a given ct_{max}. The value of ct_{max} must be less than or equal to the total task times and at the same time it also should be greater than or equivalent to the largest task time in data. Only one workstation will be required whenever the value of ct_{max} is exceed or the same as total task times. No solution will obtained as the value for ct_{max} is less than or equivalent to the largest task time in data. The respecting conditions are used for ct_{max}.

$$\max t_i \leq ct_{max} \leq \sum t_i$$
If $ct_{max} \geq \sum t_i$ then $m = 1, E = \frac{T_{total}}{1.T_{total}} = 1$ thus, Balance loss = 0
If $ct_{max} \geq \max t_i$, no solution

After the value of ct_{max} has been set, the optimal number of workstations m can be attained by using the spreadsheet. The value of m lies between m_{min} and m_{max} and it has been calculated as equation (4) and equation (5):

$$m_{min} = \left\lceil \sum_{i=1}^{n} \frac{t_i}{ct_{max}} \right\rceil \tag{4}$$

$$m_{max} = \left\lceil \frac{\sum t_i}{\max t_i} \right\rceil \tag{5}$$

where $m_{min} \leq m \leq m_{max}$

In another work, Zacharia & Nearchou minimised the number of workstations m and cycle time c using fuzzy task processing times so-called as f-SALBP-E [17]. The objective functions of the problem are to maximise the efficiency of the line, simultaneously minimising the number of workstations m and the cycle time c. The fuzzy efficiency $\check{e}$ of the line is linearly dependent with summation of fuzzy processing times of all the task $\dot{t}_{sum}$. It is also can be attained by minimising the product of number of workstations and fuzzy cycle time of the line. The line efficiency function is represented by equation (6):

$$\check{e} = \frac{\dot{t}_{sum}}{m.\check{c}} \tag{6}$$

where $\dot{t}_{sum}$: total sum of the fuzzy processing time of all the tasks
$\check{c}$: fuzzy cycle time of the line

The uncertainty and variability of task processing time and cycle time are presented by triangular fuzzy numbers (TFNs). A heuristic method based on Genetic Algorithm (GA) has been developed to solve the *f*-SALBP-E as it is a type of NP-hard optimisation problems. A two-phase GA is used for the purpose to solve the problem. In this approach, the optimal solution found from the first run is used to generate the early population of the binary run. There is no resource constraint being stated in the study. By considering the fuzzy processing time for the single assembly line balancing problem, a formulated mathematical model is performed and thus minimised the number of workstations and the fuzzy cycle time on the line.

A new genetic algorithm has been presented by Al-Hawari et al. to solve multi-objective simple assembly line balancing problem [18]. Minimisation of number of workstations, minimisation of workload variation, and maximisation of line efficiency are considered as the objective functions in the study. A Multi-Assignment Genetic Algorithm (MA-GA) has been proposed by the researcher with the combination of forward, backward, and bidirectional methods. The researcher concluded that the proposed algorithm has shown a better performance in solving multi-objective simple assembly line balancing for a larger size of problem. Equation (7) represents the line efficiency, E which is supposed to be maximised.

$$\max E = \frac{\sum_{i=1}^{n} t_i}{m.C_a} \tag{7}$$

The efficiency of the line can be maximised by minimising both variables; the actual number of workstations m and the actual cycle time of the assembly line $= \max_{1 \leq k \leq m}\{t(S_k)\}$ whereas the sum of handling time of task i is fixed. The minimum number of actual workstations m can be obtained using the mathematical formulation as stated in equation (8):

$$\min m = \sum_{k=1}^{M} \max_{1\leq i\leq n}\{x_{ik}\} \tag{8}$$

$$x_{ik} = \begin{cases} 1, & \text{if task i is assigned to station k} \\ 0 & \text{otherwise} \end{cases}$$

$$t(S_k) = \text{the total time assigned to workstation k}$$

Suwannarongsri et al. has proposed a combination of tabu search (TS) and genetic algorithm (GA) to identify the solution for simple assembly line balancing problem [13]. The goals of the problem are to (i) minimise the number of workstations, (ii) minimise the workload variance, (iii) minimise the idle time and (iv) maximise the efficiency of the line. The maximum line efficiency can be calculated by using equation (9):

$$max\ L_{eff} = max\,\frac{\sum_{i=1}^{n} T_i}{(n \times ct\_r)} \times 100 \tag{9}$$

where n : number of workstations
T_i : processing time of i^{th} workstation
ct_r : actual cycle time
L_{eff} : line efficiency

3. SALB-E optimisation algorithm

A two-part genetic algorithm (GA) is established to solve *f*-SALBP-E [17]. The first part of GA started with generating initial population, followed by performing the best solutions until it reached termination conditions. The optimal solution achieved from the first attempt is used as the source for the early population in the binary part for the purpose to find a better performance. The algorithm rises in a good feasible solution which is approximately to the exact solution in an acceptable time period.

The algorithm proposed by Al-Hawari et al. uses the combination of forward, backward, and bidirectional methods of task assignment [18]. These methods are used to assign each of tasks in a chromosome to workstations. Priority-based encoding, crossover, mutation, sequence encoding, decoding (assignment), evaluation, and selection are the primary procedures in MA-GA. As mentioned previously, the researcher simplified that the proposed MA-GA can solve problem for a larger size. It provides many feasible solutions of task assignments by combining the three methods simultaneously instead of combine using the only forward method. MA-GA will also increase the probability of identifying the optimal solution.

Suwannarongsri et al. used TSGA-based method which is the combination of TS and GA method to find the solutions for simple assembly line balancing problem. The researchers have performed a test of all type of SALBP problems from a literature against the proposed method. The result showed that the proposed TSGA-based method is capable in producing better solutions compared with conventional method [13].

Most of previous researcher used genetic algorithms (GAs) as an optimisation technique especially in SALB problem [2, 11, 12, 19, 20]. However, only a small number of researchers are focusing on simple assembly line balancing of Type-E problem [13, 14, 17]. As a consequence, the implementation of GA method has not been widely publicised in SALB-E itself.

4. Genetic algorithm for SALB-E

Genetic Algorithms (GAs) are mainly used by researcher for optimising large and complex problem specifically in SALB problem [2, 21-24]. GAs used a direct random search as an optimisation method for complex problem with the aim of finding optimum solutions [21]. The application of genetic algorithm is quite popular compare with the simulated annealing and ant colony optimisation [25].

In [17], the design of GA comprises of chromosome's encoding, a decoding mechanism, an evaluation mechanism, generation of early population, and generation of offspring. The solution for *f*-SALBP-E is characterised by chromosome's encoding, which is consists of tasks priorities (first part of the chromosome) and number of workstations on the line (binary part of chromosome). The tasks are then assigned to workstations by using a suitable decoding scheme. In evaluation mechanism, an individual chromosome with higher fitness value tends to have higher probability to be selected. The feasible tasks provide a better solution for the problem as it has low values of total fuzzy idle time.

The early random population undergoes selection, crossover, and mutation process to produce new generation. The optimum solution obtained from the first part is used as the source for the early population in second part for the aim of finding a better solution. A roulette wheel method is used in selection process. Chromosomes with higher fitness value will be selected to produce new population. Crossover operator is developed to produce new chromosomes from two parents' chromosomes by changing the tasks order. In GA, mutation mechanism worked by flipping or swapping an only chromosome to produce a single new chromosome.

Previous paper presented by Al-Hawari et al. used three assignment methods (i) forward (ii) backward and (iii) bidirectional in Multi-Assignment Genetic Algorithm (MA-GA) [18]. A forward assignment method is the mainly used for solving SALBP. By using this technique, the works are allocated sequentially to workstations by taking into consideration the cycle time constraint. In backward assignment, a flipping method is used. The task sequence chromosome is flipped to be assigned using forward assignment method whereas, the bidirectional assignment method used both forward and backward directions. From the acquired result, bidirectional assignment attained the best solution.

Three genetic operators that have been used in GA are (i) crossover (ii) mutation and (iii) selection. The researcher used weight mapping crossover operator (WMX), swap mutation operator, and roulette wheel selection (RWS). The crossover operates two chromosomes (parent) to produce a new chromosome. One-point WMX is used in the proposed MA-GA and one crossover cut has been pointed at anyplace along the length of the parent, producing two offspring that have their genes. In the research, the swap mutation operator is used in order to keep the genetic diversity. In selection step, the roulette wheel selection method has been applied to produce a new population. The chromosomes with higher fitness value get more chances to be selected. To avoid the loss of the best chromosome(s), an elitism approach is adopted while using the RWS.

Suwannarongsri et al. used TS method to determine the number of tasks assign in each workstation whereas GA is employed to assign the sequence of tasks for each workstation by considering the precedence constraints [13]. The searching process of the GA is comparable to the nature development of biological beings. The flowchart of GA is summarised as in figure 2.

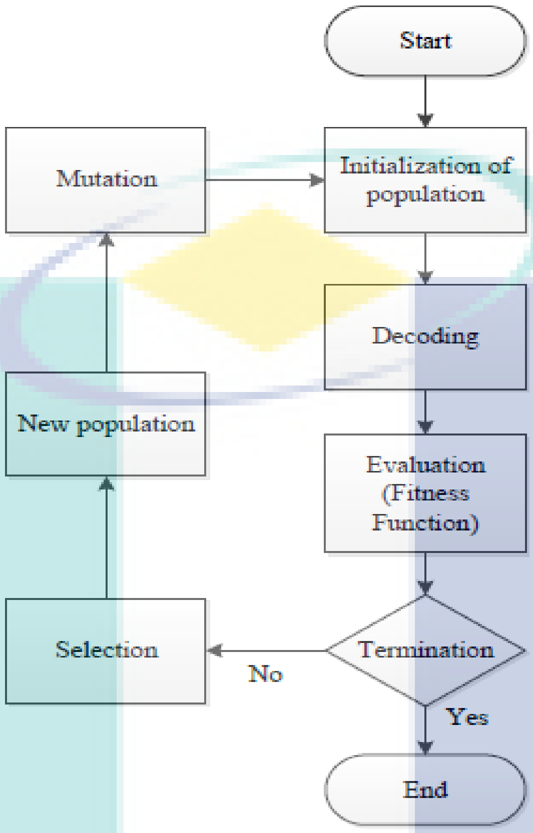


Figure 2. Flowchart of genetic algorithm.

5. Conclusion

This paper reviewed the optimisation algorithm and techniques used by the previous researcher on SALB-E. From literature review that have been conducted, it can be concluded that the application of genetic algorithm (GA) as an optimisation technique are on the rise due to its ability to solve a large-scale optimisation problem as well as searching near optimal solution.

Only a few studies are focusing on SALB-E as it is a general and complex problem. Up till now, none of them are concern on the resource constraint in the problem especially machine and tool constraint. Future research direction could be to consider recourse constraint in the optimisation of SALB-E itself.

Acknowledgement

This study was supported by a research grant from the Ministry of Higher Education under RDU140103. The authors would like to thank Universiti Malaysia Pahang for the facilities provided.

References

[1] Rashid MFF, Hutabarat W, Tiwari A. A review on assembly sequence planning and assembly line balancing optimisation using soft computing approaches. The International Journal of Advanced Manufacturing Technology. 2012;59:335-349.

[2] Gu L, Hennequin S, Sava A, Xie X. Assembly line balancing problems solved by estimation of distribution. Automation Science and Engineering, 2007 CASE 2007 IEEE International Conference on: IEEE; 2007. p. 123-127.

[3] Kriengkorakot N, Pianthong N. The Assembly Line Balancing Problem. KKU Enginieering Journal. 2007;34:133-140.
[4] Nearchou AC. Multi-objective balancing of assembly lines by population heuristics. International Journal of Production Research. 2008;46:2275-2297.
[5] Emeke Great O, Offiong A. Productivity Improvement In Breweries Through Line Balancing Using Heuristic Method. International Journal of Engineering Science & Technology. 2013;5.
[6] Scholl A, Becker C. State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. European Journal of Operational Research. 2006;168:666-693.
[7] Boysen N, Fliedner M, Scholl A. A classification of assembly line balancing problems. European Journal of Operational Research. 2007;183:674-693.
[8] Becker C, Scholl A. A survey on problems and methods in generalized assembly line balancing. European Journal of Operational Research. 2006;168:694-715.
[9] Chong KE, Omar MK, Bakar NA. Solving assembly line balancing problem using genetic algorithm with heuristics-treated initial population. 2008.
[10] Ponnambalam S, Aravindan P, Naidu GM. A multi-objective genetic algorithm for solving assembly line balancing problem. The International Journal of Advanced Manufacturing Technology. 2000;16:341-352.
[11] Zhang W, Gen M, Lin L. A multiobjective genetic algorithm for assembly line balancing problem with worker allocation. Systems, Man and Cybernetics, 2008 SMC 2008 IEEE International Conference on: IEEE; 2008. p. 3026-3033.
[12] RuiJun Z, DingFang C, Yong W, ZhongHua Y, Xinxin W. Study on line balancing problem based on improved genetic algorithms. Wireless Communications, Networking and Mobile Computing, 2007 WiCom 2007 International Conference on: IEEE; 2007. p. 2033-2036.
[13] Suwannarongsri S, Limnararat S, Puangdownreong D. A new hybrid intelligent method for assembly line balancing. Industrial Engineering and Engineering Management, 2007 IEEE International Conference on: IEEE; 2007. p. 1115-1119.
[14] Gurevsky E, Battaïa O, Dolgui A. Balancing of simple assembly lines under variations of task processing times. Annals of Operations Research. 2012;201:265-286.
[15] Suwannarongsri S, Puangdownreong D. Multi-objective assembly line balancing via adaptive tabu search method with partial random permutation technique. Industrial Engineering and Engineering Management, 2008 IEEM 2008 IEEE International Conference on: IEEE; 2008. p. 312-316.
[16] Wei N-C, Chao I-M. A solution procedure for type E simple assembly line balancing problem. Computers & Industrial Engineering. 2011;61:824-830.
[17] Zacharia PT, Nearchou AC. A meta-heuristic algorithm for the fuzzy assembly line balancing type-E problem. Computers & Operations Research. 2013;40:3033-3044.
[18] Al-Hawari T, Ali M, Al-Araidah O, Mumani A. Development of a genetic algorithm for multi-objective assembly line balancing using multiple assignment approach. The International Journal of Advanced Manufacturing Technology. 2014:1-14.
[19] Gonçalves JF, De Almeida JR. A hybrid genetic algorithm for assembly line balancing. Journal of Heuristics. 2002;8:629-642.
[20] Sabuncuoglu I, Erel E, Tanyer M. Assembly line balancing using genetic algorithms. Journal of intelligent manufacturing. 2000;11:295-310.

[21] Ranjan R, Pawar P. Assembly Line Balancing Using Real Coded Genetic Algorithm. International Journal of Scientific Research in Computer Science and Engineering. 2014;2:1-5.
[22] Matondang MZ, Jambak MI. Soft computing in optimizing assembly lines balancing. Journal of Computer Science. 2010;6:141.
[23] Tasan SO, Tunali S. A review of the current applications of genetic algorithms in assembly line balancing. Journal of intelligent manufacturing. 2007;19:49-69.
[24] Mohd Razali N, Geraghty J. Biologically inspired genetic algorithm to minimize idle time of the assembly line balancing. Nature and Biologically Inspired Computing (NaBIC), 2011 Third World Congress on: IEEE; 2011. p. 105-110.
[25] Battaïa O, Dolgui A. A taxonomy of line balancing problems and their solutionapproaches. International Journal of Production Economics. 2013;142:259-277.

Int J Adv Manuf Technol
DOI 10.1007/s00170-016-9158-3

ORIGINAL ARTICLE

A review of two-sided assembly line balancing problem

Muhammad Razif Abdullah Make[1] · Mohd Fadzil Faisae Ab. Rashid[1] · Muhamad Magffierah Razali[1]

Received: 27 March 2016 / Accepted: 8 July 2016

Abstract Assembly line balancing (ALB) is concerned with assigning tasks within an assembly line to meet the required production rate for optimization purposes. On the other hand, two-sided ALB performs double-sided assembly operation on a single assembly line. In this paper, we have focused the survey on two-sided assembly line balancing (2S-ALB) research problems. The numerous factors mentioned in 2S-ALB literature were actually based on problem resolutions, and this paper will quote any preferred literature considering the frequent citation. In particular, this review explores in detail the ALB problems, optimization methods, objective functions, and specific constraints used in solving 2S-ALB problems. Among the purposes of ALB problems is that it traditionally focuses on simple ALB with various engaging approaches. General ALB comes second because of its complexity and nondeterministic polynomial (NP)-hard-classified problems. However, due to the current manufacturing issues, GALB problems, such as 2S-ALB, are forced to be examined and this comprehensive literature will specify anything necessary for the optimization purposes. Finally, future research direction has been discovered and put forward as the suggestion.

Keywords Assembly line balancing · Two-sided · Artificial intelligence

✉ Mohd Fadzil Faisae Ab. Rashid
ffaisae@ump.edu.my

1 Manufacturing Focus Group, Faculty of Mechanical Engineering, Universiti Malaysia Pahang, 26600 Pekan, Pahang, Malaysia

1 Introduction

In a modern manufacturing system, assembly line balancing (ALB) plays a vital function, especially in the production line. The installation of an assembly line is a long-term decision and requires large capital investments. It is important that such a system is designed and balanced so that it is able to work as efficiently as possible [1–3]. The assembly line was introduced by Henry Ford in his automobile plants. Since then, many developments through researches have been introduced [4].

Generally, from the feature of the product and technical operational requirement, there have been differences in the line balancing problem classifications made by the researchers. For instance, [5] classified the line balancing problems into simple and general types of problems. The similar classification was also used by [6, 7]. Besides that, the line balancing also was classified according to the model number (single-model and multimodel) and the nature of task times (deterministic and stochastic) by [8–10]. On the other hand, [11, 12] classified the line balancing problems into two types: one-sided and two-sided ALB problems.

Both two types of assembly lines are quite famous among researchers. One-sided assembly, or commonly called single-sided assembly line, was examined extensively in the past few decades. The assembly line is a flow-line production system in which a series of stations are arranged along a conveyor belt or a similar mechanical material handling system [13]. The stations are often prepared in a single line which is long enough to complete the desired product with different types of tasks or assembly processes, as illustrated in Fig. 1. Frequently, every station only has one operator to manage each task and fully run the assembly line. The operator cannot leave the station when the assembly process is running.

Although the focus of researches are always on one-sided assembly lines, two-sided assembly lines are recognized to be

Published online: 02 August 2016

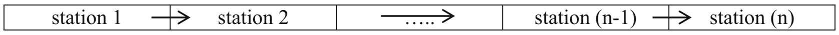


Fig. 1 Single-sided assembly line

crucially important too, especially in the assembly of the large-sized products like cars, busses, or trucks [14]. Two-sided assembly line or often called the double-sided assembly line is absolutely different compared to the one-sided or single-sided assembly line. In the two-sided assembly lines, the operating direction of the assembly tasks will be carried out on the same product in parallel at both the left and right sides of the lines. Due to the use of both sides of the lines, the tasks will have additional operating direction restrictions. The directions can be classified into three types: the left side (L), the right side (R), and either side (E) [13, 15, 16]. Figure 2 illustrates the example of a two-sided assembly line. One of the main differences between the single- and two-sided assemblies is the restriction on the operating directions. Some of the assembly operations can be performed at only one of the sides, while others can be performed on either side of the lines [17]. The two-sided assembly line used both of the lines to enhance the assembly performance of a complex production system such as in automotive industries.

In industry, line balancing is very important in taking advantage of them. Unbalanced lines may incur unnecessary cost [4]. Hence, ALB was raised among researchers in order to satisfy the workload and increase the operational line efficiency [18]. The activity of balancing operations has appeared since 1955 [4]. In order to increase line efficiency, the balancing operations are responsible to determine the set of tasks. ALB is generally classified into two, either simple assembly line balancing (SALB) or general assembly line balancing (GALB) [6, 10]. Figure 3 illustrates the ALB classification and problem-type examples.

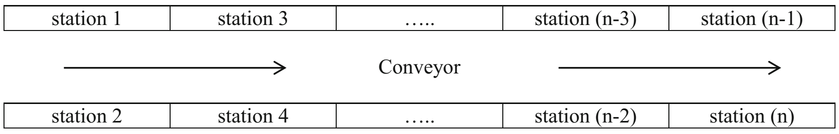
Normally, SALB studies only on a single side of the assembly line (Fig. 1). Even so, SALB problems have been categorized into some classes. First, SALB in type 1 (SALB-1) will perform the minimization of workstation numbers for a given cycle time as the objective [19, 20]. Second, type 2 assembly will consider the minimization of cycle time with the given number of workstations [19, 21]. Next is SALB for type E problems which is different in line configuration from the single-sided ALB [22–25]. This SALB-E type is significantly believed to have its own advantage. Another class in SALB is type F that was categorized by Kriengkorakot and Pianthong (2007) in their studies [10]. SALB type F also has been discussed in the Assembly Line Balancing book by Micieta B. and Stollmann V. (2011) [26].

On the other hand, another ALB class also has been justified in general form (GALB) [27]. First was by the two-sided assembly line balancing (2S-ALB) [28–31], followed by the mixed model of ALB (MALB) [4, 32–35], and then the U-type of ALB [36–40]. The GALB of 2S-ALB will be discussed in detail in another section on the ALB problems. The MALB problem is normally for high production with multiple types of products [41]. However, in GALB, there are many other types of assemblies that have been introduced by other researchers [42]. It was also included with combinatorial problem types. The GALB is always considered as the nondeterministic polynomial (NP)-hard problem due to its complexity.

The development of 2S-ALB in GALB seems more crucial over single-sided assemblies. Two-sided assembly lines were introduced in 1993 by Bartholdi [43], who conducted an iterative program with balancing algorithm using the first fit heuristic. Then, consequently, it was continued by other researchers [44–46], with genetic algorithm as the solving approach. Meanwhile, [47] has proposed an ant colony algorithm to solve two assembly line balancing problems focusing on the minimization of workstations and the maximization of work relatedness. In addition, the simulated annealing for a two-sided assembly problem are successfully presented in [48, 49] and again for mixed-model two-sided assembly line balancing problem in [50].

In previous conducted research, the objective function has also been given considerable attention. In [4, 13, 46, 50–52], the minimizing number of workstations and mated stations are selected as the applied objectives. Besides, [15, 36, 53, 54] have adopted the minimizing number of workstation and line length as their preferred objective function. Apart from that, multiobjective function is also presented in studies by [16, 29, 48, 55, 56]. A continuous evaluation toward 2S-ALB constraint for single [17, 30, 46, 47, 56] and multiple considered constraints [29, 49] is addressed very well. This paper reviews on 2S-ALB problems to address problem types, optimization methods, objective functions, and considered constraints. Many optimization problems have been successfully studied by other researchers. Hence, this review will discuss through their literature and studies.

Fig. 2 Two-sided assembly line



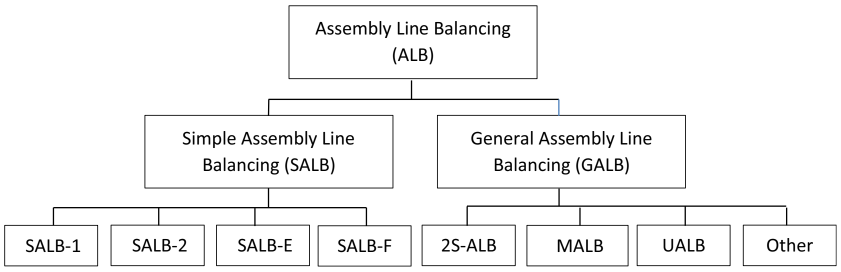

Assembly Line Balancing (ALB)
Simple Assembly Line Balancing (SALB)
General Assembly Line Balancing (GALB)
SALB-1
SALB-2
SALB-E
SALB-F
2S-ALB
MALB
UALB
Other

Fig. 3 ALB classification

2 Assembly line balancing problem

The assembly line balancing problem (ALBP) was first mathematically formulated by Salveson in 1955 [26, 29]. ALBP is the problem of assigning tasks to stations in such a way that one or more objectives are optimized, subjected to some specific constraints. Since then, many researches on assembly lines have included the exact solution methods, heuristics, and metaheuristic approaches reported in the literature. The heuristic method is actually an answering method to produce a solution that applies the trial-and-error strategy with a reasonable functional period [57]. On the other hand, the metaheuristic approach is an independent solution strategy that is believed to resolve any optimization problem. The ALBP assigns tasks in an ordered fashion to every workstation by satisfying specific constraints [58–60]. The related studies on ALB are classified in various types of problems. Since this study considers 2S-ALB as the main problem, the other ALB problems will also be discussed.

2.1 General two-sided assembly line balancing

Two-sided assembly line was introduced for the first time by Bartholdi in 1993 [16, 59] to produce high volumes of large-sized products. Typically, the 2S-ALB production of bus and trucks by Kim, Kim, and Kim in 2000 [44], automobile by Lee et al. in 2001 [11], and a domestic product by Baykasoglu and Dereli (2008) [47] studied on the problem to find the best solution. Since then, numerous researches have brought up different methods, either heuristic or metaheuristic as the solution approach.

As shown in Fig. 2, 2S-ALB generally has a pair of workstations/stations facing each other almost in all operation lines. A pair of stations facing each other, e.g., station 1 and station 2, is called "mated station," and one of the stations is called "companion" [44]. Every operating station will perform a different task despite the stations being opposite of each other. Large assembly production industries such as cars and trucks need this type of assembly lines for them to perform the assembly operation at a given time to achieve their productivity target. The operational process in 2S-ALB will provide several advantages compared to the single-sided ALB [14].

The comparisons between 2S-ALB and single-sided ALB are remarkably different. As indicated earlier, 2S-ALB has a more complex layout against single-sided ALB. Figure 4 illustrates the 2S-ALB layout with the left and right sides of the workstation. The number in the boxes indicates the task on every side of the workstation. For a single-sided ALB, the sides are definitely not crucially tough, because it only has a single-side layout of the operation line [29, 61]. In addition, the task distribution between the compared ALB problems differs as well. Having two preferred sides of the assembly line, 2S-ALB definitely needs to distribute every task accordingly. For a single-sided assembly, the precedence relation is considered appropriate with all the tasks assigned to a workstation that can be carried out without any interruption. However, its difference from 2S-ALB is that some of the tasks assigned could be delayed after the assigned task of its companion [30, 43], or commonly known as idle time [48]. The shaded area in Fig. 4 denotes the idle time which is unavoidable in completing the 2S-ALB processing product.

Throughout the 2S-ALB installation, it will lead to some valuable advantages of the assembly lines [45], such as the following:

1. Shortens the assembly line
2. Saves some spaces on the assembly lines
3. Reduces the cost of tools and fixtures
4. Reduces the throughput time
5. Reduces material handling

Based on these advantages, it is considered to have the ability to maximize the productivity of the assembly lines. In reality, high attention has been given for the study of 2S-ALB

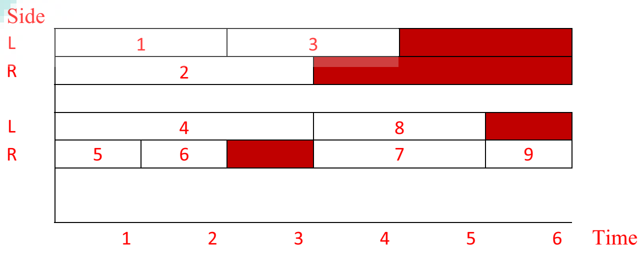


Fig. 4 2S-ALB task distribution layout

problems. Two decades have passed since it was firstly introduced by Bartholdi for his large volumes of vehicle production [43]. Initially, the aim is to simplify and facilitate the manufacturing industries of the 2S-ALB functional capabilities. Up until now, the studies focusing on 2S-ALB problems continue to attract participation among public researchers. Nowadays, 2S-ALB has also been implemented in other manufacturing industries, for instance, in furniture and electrical appliance production [4]. Moreover, the 2S-ALB problem complexity is also growing day after day, with the combination and hybridization of the ALB problems. Therefore, currently, better performance of optimization algorithm is favorably built by researchers in line with the 2S-ALB progress. Besides, introducing and emphasizing a new framework for 2S-ALB problem solutions are also involved.

In 2S-ALB studies, researchers have stated and applied many different kinds of approaches successfully. For example, Purnomo and Wee (2014) [30] proposed the harmony search method to solve 2S-ALB problems. On the other hand, in the [44–46] studies, the genetic algorithm method was performed. The simulated annealing algorithm was also applied in some other studies [48, 49]. Besides, the heuristic approach, which is a problem-dependent method, has been addressed as well for 2S-ALB problems [11, 47, 51, 62].

2.2 Two-sided mixed-model assembly line balancing

The mixed-model line balancing problem was first introduced by Thomopoulos in 1967 [55]. Considering today's competitive market, the MALB has become more advantageous rather compared with a single model assembly line balancing. A single model assembly line only designs a single standardized homogenous product, while the mixed-model assembly lines are widely applied to produce two or more product models depending on the customer needs [4]. In other words, a mixed-model assembly line is designed to produce a similar set of products in the different mixed-ordered model.

The existing research for the MALB problem addressed single- and multi-objective problems under various assembly line considerations. Commonly, in configuring a mixed-model assembly line, a lot of goals and objectives are considered. There are two goals that have been studied by most researchers [63] in balancing the mixed-model assembly lines:

1. Leveling workloads for every station on the line
2. Leveling part usage on the line

The first goal of leveling the workload for all stations on the line is attempting to achieve a balanced workload at specific times for each assembly task, while the second goal is attempting to minimize the variation used by the different parts over time.

In order to fill customer requirement, MALB is applied widely in a range of industries; for instance, in the production of electrical appliances, furniture, and clothing [4]. In automotive industries, the mixed-model assembly line was broadly introduced in combination with the two-sided assembly line. For example, [4, 14, 16, 50] studied a two-sided mixed-model assembly line balancing established with a different solution balancing approach. These optimally gave a positive effect on the large-sized high-volume production industries such as in automobile and appliance factories.

2.3 Two-sided parallel assembly line balancing

The parallel line configuration idea in ALB was started by Suer and Dagli in 1994 [59]. The combination of two or more lines placed parallel to each other became an idea of sharing tools and fixtures to complete an entire job. The balancing idea of P-ALB was studied by Gökçen, Agpak, and Benzer in 2006 [64] with the title Balancing of Parallel Assembly Lines. They proposed a new procedure with a mathematical model on the single-model assembly line balancing problem with parallel lines. Since then, the researcher broadly continued the study on the P-ALB problem. Various approaches and ideas to solve the P-ALB problem then emerged. Cercioglu, Ozcan, Gokcen, and Toklu (2009) proposed a simulated annealing approach for solving the P-ALB [65]. Meanwhile, Ozcan, Cercioglu, Gokcen, and Toklu (2009) firstly utilized a multiobjective Tabu search algorithm method on parallel assembly lines [66]. A novel ant colony optimization (ACO-based algorithm also became one of the methods for solving the P-ALB problems by Baykasoglu, Ozbakir, Gorkemli, and Gorkemli in 2009 [59, 60].

Parallel assembly lines, usually built with two or more lines, are located parallel to each other. This provides the following advantages [64] to the lines:

1. Shortens the assembly lines
2. Steadily runs during breakdown

By installing parallel configuration of the assembly lines, it definitely shortens the assembly lines. Besides, being able to locate only one operator in between the adjacent station helps the operator to perform both tasks. These completely utilize the workers on the assembly lines [60]. Another advantage of parallel assembly lines is that it could still be run steadily even when a workstation faces a problem or breakdown [64]. A single assembly line will stop the assembly operation if any workstation faces a problem, but P-ALB will continue to run and perform the task at the other adjacent lines. The advantages of parallel assembly lines over a single assembly line were also discussed by Ozcan, Gokcen, and Toklu (2010) [66]. It is able to provide much more benefits:

1. It can help to produce similar products or different models of the same production of the adjacent lines.
2. It can reduce the idle time and increase the efficiency of the assembly lines.
3. It is able to make production with a different cycle time for each of the lines.
4. It can improve visibility and communication skills between operators.
5. It is also able to reduce operator requirements.

The combination types of production lines for the parallel and two-sided lines were already studied. The parallel two-sided assembly line balancing problem (PTALBP) was firstly developed by Ozcan, Gokcen, and Toklu in 2010 [66], focusing on the large-sized productions in different industries. Other studies on the PTALB problem were reviewed by Ağpak and Zolfaghari (2015) [53] and Kucukkoc and Zhang (2015) [60].

3 Optimization method

Since 1955 by Salveson, various researches regarding the ALB solving problem were introduced [27]. The researchers focused on improving the assembly line, so that it was able to work efficiently. In 1993, Bartholdi first presented his idea to address the 2S-ALBPs. He discussed some theoretical properties of the 2S-ALB and proposed the first-fit heuristic algorithm method of assigning tasks to workstations [15, 30]. Since then, numerous researches concerning the ALB solution problems using different methods have been introduced. The mathematical model and heuristic and metaheuristic methods were developed to solve the ALB for different problem types. Table 1 summarizes the optimization method used in previous researches of the 2S-ALB problems. Meanwhile, Fig. 5 below shows the number of research papers that have successfully implemented the different types of algorithm by using different soft computational methods.

The number under the graph (Fig. 5) represents the different types of optimization in the 2S-ALB research paper (see Table 1 legend). Mostly all of these metaheuristic algorithm methods were inspired by natural phenomena. Among them, the effectiveness in solving the NP-hard optimization problem became necessarily important. The ALB problem type became complex day after day and the high capability of the algorithm method seems more needed. From the survey, the most (metaheuristic) frequent optimization algorithms used are genetic algorithm (GA) and ant colony optimization (ACO) algorithm (used five times from 30 papers) followed by simulated annealing (SA). The high applied value of GA and ACO in Fig. 5 shows the popularity and stability of these methods in solving the 2S-ALB problems. Others might be less studied because the relatively new algorithm and efficiency of the method was not well proven yet.

3.1 Genetic algorithm

Genetic algorithm was formally introduced in 1970s in the University of Michigan by John Holland. GA has been proven to be very efficient and powerful in a wide variety of applications [44]. It provides a method to find the best sequence of assembly process among the possible sequences that have been generated either in constrained or unconstrained condition. GA is also considered as one of the artificial algorithm methods or artificial intelligence-based algorithms in solving the ALB problems. The accomplishment of GA in solving difficult and complex combinatorial problems is seen to have outperformed the other algorithms in terms of solution quality and convergence speed [46]. Genetic algorithm is believed to be able to find the optimal or nearly optimal assembly plans for the model structure generated by analyzing the small number of possible solutions.

Algorithm starts with a set of solutions (represented by chromosomes) called population. Solutions from one population are taken and used to form a new population [32]. This is motivated by a hope that the new population will be better than the previous one. Solutions which are selected to form new solutions (offspring) are selected according to their fitness—the more suitable they are, the more chances they have to reproduce. This process is repeated until some conditions (improvement of the best solution) are satisfied.

In solving 2S-ALB problems, many researchers have used the GA method [12, 44–46, 60]. The reputation of GA was first addressed by Kim and Kim et al. [44, 46] in solving the 2S-ALB problems. They successfully implemented the GA method with the objective of minimizing the number of stations with a given cycle time. In 2009, Song et al. used a mathematical model and GA for the 2S-ALB problem with different objectives of minimizing the cycle time [45]. Both studies have inspired other researchers to implement GA in solving other types of balancing problems in assemblies.

Many compliments and praises were given to the performance of the GA method in solving different kinds of complex combinatorial problems nowadays [32]. However, some weaknesses arise since GA has been used in the ALB problems [6]. The premature convergence turned into an issue [68, 69]. This appears to be due to that GA sequences heavily depend on the initial generating sequence. Besides, it requires a high amount of computational time in order to find the final solution [70]. Conversely, in a study by [71], the GA method behavior has been discussed to greatly depend on numerous control parameters and only used simple test data. The disadvantage and weaknesses of the GA method should be considered for future research direction.

Table 1 Method of optimization for 2S-ALB problems

Author/s, Year	Ref.	Optimization method												
		1	2	3	4	5	6	7	8	9	10	11	12	13
Yuan, Zhang et al. 2015	[4]				x									
Kucukkoc and Zhang 2015	[59]		x											
Kucukkoc and Zhang 2015	[60]	x												
Chiang, Urban et al. 2015	[15]					x								
Ağpak and Zolfaghari 2015	[53]													x
Tuncel and Aydin 2014	[29]							x						
Purnomo and Wee 2014	[30]						x							
Kucukkoc and Zhang 2014	[55]		x											
Kucukkoc and Zhang 2014	[54]		x											
Chutima and Naruemitwong 2014	[56]								x					
Purnomo, Wee et al. 2013	[12]	x												
Khorasanian, Hejazi et al. 2013	[48]			x										
Tapkan, Ozbakir et al. 2012	[61]				x									
Roshani, Fattahi et al. 2012	[49]			x										
Chutima and Chimklai 2012	[16]					x								
Ağpak, Yegül et al. 2012	[36]									x				
Taha, El-Kharbotly et al. 2011	[46]	x												
Özbakır and Tapkan 2011	[17]				x									
Xiaofeng, Erfei et al. 2010	[67]										x			
Özcan and Toklu 2010	[51]													x
Özcan, Gökçen et al. 2010	[66]											x		
Özcan 2010	[13]			x										
Simaria and Vilarinho 2009	[14]		x											
Özcan and Toklu 2009	[50]			x										
Özcan and Toklu 2009	[52]												x	
Kim, Song et al. 2009	[45]	x												
Hu, Wu et al. 2008	[62]													x
Baykasoglu and Dereli 2008	[47]		x											
Lee, Kim et al. 2001	[11]													x
Kim, Kim et al. 2000	[44]	x												

Optimization method: 1—genetic algorithm, 2—ant colony optimization, 3—simulated annealing, 4—bee algorithm, 5—particle swarm optimization, 6—harmony search, 7—teaching learning based optimization, 8—Pareto biogeography based optimization, 9—lexicographic optimization method, 10—branch and bound, 11—Tabu search, 12—goal and fuzzy goal, 13—other heuristic methods

3.2 Ant colony optimization

Ant colony algorithm is one of the most famous metaheuristic methods that have already been used successfully for solving various problems in ALB. It was introduced in the early 1990s by Marco Dorigo [72]. The ACO algorithm method studied by [6] was also considered to have high reputation following under the GA fame in solving many types of ALB problems. It

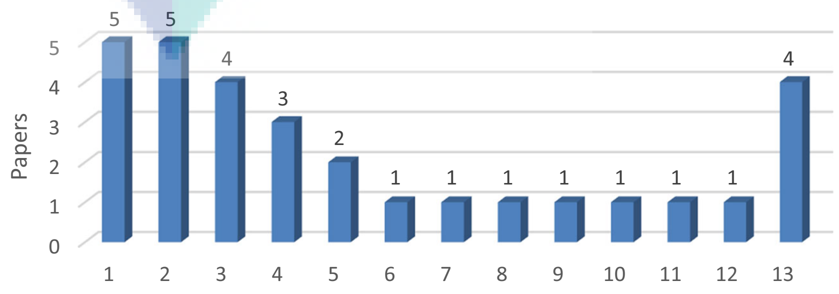


Fig. 5 Number of papers applied different algorithm methods in 2S-ALB

was already assessed to be fit in overcoming and solving even in high combination problems.

The ACO algorithm was inspired by the behavior of a real ant colony finding a path between the food source and its nest. The pheromone trail released by the other ants will be followed. Each ant from the colony will come out with a different path. The ants which pick the shortest path will return to the nest faster; hence, there will be much more pheromone trail on the shortest path. It influences other ants to follow that path [54]. The pheromone trail of an ant path was considered a solution in the algorithm, and the performance will be evaluated according to its quality of accomplishment in obtaining the final execution or solution [55].

Previous studies that successfully presented and used the ACO method from the literature provided essential trend in solving the 2S-ALB problems. The study by Baykasoglu and Dereli (2008) [47] followed by Simaria and Vilarinho (2009) [14] presented the successful achievement in balancing the 2S-ALB problems. While in 2014, Kucukkoc and Zhang became the first pioneer to address the ACO method through the mixed-model parallel two-sided assembly line balancing problem with model variations [54, 55]. They were practically successful in implementing this algorithm method into large-sized products. Then, in 2015, the knowledge of the type-E parallel two-sided assembly line balancing problem was introduced [59] for the first time in literature by Kucukkoc and Zhang in their research; type-E parallel two-sided assembly line balancing problem: Mathematical model and ant colony optimization-based approach with optimized parameters.

From the previous published literature, the ACO algorithm contributed to be competitive in solving different kinds of ALB problems, despite its strong global search ability. However, this evolutionary algorithm also holds its own weakness and limit. For example, the pheromone trail path made by the ant always evaporates and disappears if the path is bad [6]. Therefore, it will also cause premature convergence. Nevertheless, the premature convergence in the ACO algorithm method has been solved by [73].

3.3 Simulated annealing

Numerous studies on ALB performed different approaches on the optimization method for solving the assembly problems. The heuristic, metaheuristic, and also exact approach solutions were introduced and have been reported in literature. The SA algorithm became one of the leading metaheuristic approaches in solving multiple cases or problems of ALB. It was first applied by Kirkpatrick et al. in 1983 in solving a combinatorial optimization [13].

The simulated annealing algorithm was originally inspired by the annealing process in metal works [74]. The heating and cooling process were involved against the material to alter the physical properties due to the changes in the internal structure. In the SA optimization method, it was initially set high and then allowed to cool slowly. The chance of accepting solutions actually gives the algorithm the ability to find early execution before generating the optimal solution.

The simulated annealing algorithm in solving the ALB problem is currently studied by many researchers. The review of such study was given by [13, 48–50]. They successfully implemented the SA method in their studies in minimizing or maximizing something through their objectives. The SA method provides several advantages in the ALB such as the reasonable computational time and good performance in determining the optimal solution on every sized problem [49]. This has outperformed other methods in terms of solution quality.

However, this iterative random search technique (SA) also has its weakness. The SA method is believed to be able to jump into a local optimal solution by accepting the bad solution [13, 50]. This condition will create an opportunity for the bad solution to be selected as the optimal and final solution. Other drawbacks in the SA method are stated as follows:

1. The procedure method will stop when the stopping criterion is reached in getting the optimal solution.
2. The initial solution starts with low solution value.

The two above drawbacks need a proper attention with respect to the 2S-ALB problems and for getting the optimal solution.

3.4 Other optimization methods

Besides the three optimization methods discussed earlier, there are other metaheuristic approaches used by researchers in solving the 2S-ALB problems. These algorithms are the hybrid honey bee mating optimization (HHBMO) algorithm [4], bee algorithm (BA) [17, 61], particle swarm optimization (PSO) [15], teaching learning based optimization (TLBO) [29], harmony search (HS) [30], Pareto biogeography-based optimization (PBBO) [56], particle swarm optimization with negative knowledge (PSONK) [16], lexicographic optimization method (LOM) [36], branch and bound (B&B) [67], Tabu search (TS) [33, 66], and goal and fuzzy goal programming (G&FG) [52]. Otherwise, in some researches, they applied the heuristic (problem-dependent) approach such as in [11, 47, 51, 53, 62].

As far as the search methods are concerned with the popularity efficiency, other optimization methods become neglected. However, this is different with the GA and ACO algorithm methods. They were introduced more than decades ago and the performance of optimization in various kinds of ALB problems are well known. The optimization algorithm recognition is basically based on the performance efficiency and robustness in solving differently sized (small, medium, and large) problems. Hence, it requires a long time for researchers to find and test other methods that are considered relatively new. Nevertheless, the evolutionary combination of the optimization method seems to be able to raise the new algorithm to be getting highlighted through better performance.

3.5 Comparison of different optimization methods

Among the previous researches, the optimization method in ALB strongly gives an impact to the industries. Different methods of optimization either heuristic or metaheuristic successfully develop prior to each research study. In ALB problems, GA, ACO, SA, and other relatively new algorithm optimization methods definitely attempt to balance the assembly line with high values of line efficiency. However, all those optimization methods already serve with their own advantages and some weaknesses.

A successful GA method has been recently presented with a complex combinatorial problem with more numbers of studies possessing the searching method ability. It does not require examination of all the possible solutions but uniquely, it is still able to obtain the best feasible result [6]. In [75], GA is also believed to be able to handle complex and multiple constraint problems very well, even though the premature converge [76, 77] and high amount of computational time [70, 78] became an issue. Therefore, another study has been developed to overcome the raised issues by introducing dynamic partitioning (DPa) in chromosome [76] and the combination with other soft computing algorithms [68, 69]. Kucukkoc and Zhang [60] successfully compared the obtained result with the result of Gökçen et al. [66]. By this, they have obtained a very encouraging performance as shown by GA.

Meanwhile, similar to the ACO method, it also contributes in solving various kinds of discretized ALB problems. Besides, this method is also believed to directly be able to present in a completed ACO graph [6, 79]. Furthermore, the ACO method also appears as the maximum citation paper after GA in five applied journals (Fig. 5). Despite the ACO sensation, it also comes with some confusion. In [80], a premature convergence is stated as a drawback when implementing the rule of the ACO method. For this reason, [73] have introduced a summation updating the rule to overcome this matter. Besides, an adopted particle swarm updating position has also succeeded in solving this outcome matter [80]. The hybridization of ACO and PSO method is able to solve the premature convergence and then significantly reduce the computational time. Baykasoglu et al. [81] have proposed a novel ant colony optimization-based algorithm for PALBP. They compared their test results with three other existing approaches from the literature to prove the efficiency of the proposed algorithm.

This forward to the SA optimization method presented on 1983 which is the reasonable period of computational time being recognized greatly in ALB optimization. This method outperforms the other methods by allowing faster solving solution even for a larger problem [49]. The high reputation of the SA method is practically easy to use and extremely popular in solving practical problems such as job-shop scheduling, traveling salesman, and timetabling problem [82]. Nowadays, the SA method is frequently compared with GA, besides hybridization of these two optimization methods. The main aim of hybridization is to avoid being trapped by a local minima and to have faster convergence. By this, the advantage of both methods could be developed [83]. In Cercioglu et al. [65], a simulated annealing approach in solving the PALBP is proposed. A comparison between the obtained results with the existing heuristic algorithm proposed by Gökçen, Agpak, and Benzer (2006) [64] has also been reported.

Besides the three abovementioned methods, there are many other optimization methods that have successfully shown its accomplishment. Bee algorithm applied by Ozbakir and Tapkan [17] presented for balancing the 2S-AL has effectively compared the optimization result with four other research results in seven differently sized problems. Considerably, it is best to know that the GA method has performed better solutions in computational time than other approaches did including ACO. This is followed by the PSO started by Kennedy and Eberhart in 1995 [84]. Although PSO algorithm is relatively new compared with GA and ACO, this method also holds a good criterion for being selected as an optimization method. Inspired by the social behavior of birds flocking together, the PSO has a simple algorithm with a single velocity formula to evolve and less computational resource compared with GA [6]. Chutima and Chimklai [16] have compared PSO with two different optimizations and significantly showed that results by the PSO method were much better with a simple but robust algorithm performance. This means that the other new algorithms also show a favorable appearance besides those former algorithms.

4 Objective function

Objective function is the computed measure used to evaluate the performance of assembly line. It is widely used mainly in decision analysis, operation research, and optimization studies [85]. The objective function is critically important for a research mainly in the optimization study. Conversely, in the ALB optimization research, objective function becomes necessarily important. These will strictly guide researchers to keep their direction in finding the best solution to their problems. In most studies, the objective function will define the optimization problems and either the tasks or even the installation requirement setup would need to be minimized or maximized.

All the earlier studies possess their own objective through their research. Most of them have been studied and used the multiobjective function approach rather than the single objective function. Table 2 shows the objective function used in the previous researches; Fig. 6 presents the number of researches that successfully implemented the different types of objective functions in the ALB problems.

The numbers under the graph represent different types of objective function of 2S-ALB problem (see legend of Table 2). The most popular objective function in 2S-ALB problems is to minimize the number of workstations with 21 counts from 30 papers, while the minimization of the mated station number has taken the second place in the objective function popularity.

4.1 Minimizing the workstation number

During the last decade, researchers have begun to study the 2S-ALB problems recognized to be crucially important in real life. They have developed numerous techniques and assumptions to fulfill their objective function. Even in the simple assembly lines, the number of workstations has been taken into consideration and already classified into two types [53]:

Type 1:Minimizes number of workstations for a given cycle time

Type 2:Minimizes the cycle time for a given number of workstation

Some researchers believe these two classifications can be applied into other ALB problems. Özbakır and Tapkan (2011) [17] found that the bee algorithm method in 2S-ALB has taken the Type-1 group in minimizing the number of workstations into their research objective. While Özcan and Toklu (2010) [51] also considered Type-1 objective function for their heuristic approach method.

The evaluations on minimizing the workstation number were discussed in some researches. As in Kim et al. (2000) [44] study, they have successfully determined the fitness of potential solution.

$$\text{Eval} = \sum_{j\in J} WSj \tag{1}$$

where J is the set of workstations and $0, \quad \text{if } F_j = 0,\ 1, \text{if } 0 < F_j \le CT,\ 1 + \left(\dfrac{F_j}{CT} + 1\right), \text{if} F_j > CT$.

The evaluation measure (Eq. 1) intends to select more fit individual characteristics for the next generation.

The related studies that applied the objective function to minimize the number of work stations in the ALB problems have been reported in literature [4, 13–17, 36, 44, 46–48, 50–55, 59–61, 66]. However, there is one objective function that seems to be related to the above function in minimizing the workstation number (i.e., minimize mated station number) as discussed in the following section.

4.2 Minimizing the mated station number

Formally, in two-sided assembly line, there will be a pair of lines placed opposite each other such as that shown in Fig. 2. In the 2S-ALB, both sides of the lines either right or left will perform its individual task. A mated station is represented by a pair of station or workstation that faces each other [47, 48]. In some researches, it is also called as the companion [44, 47]. Therefore, the 2S-ALB minimization of mated station number is generally able to reduce the number of stations as well. Most of the researches will take the minimization of station number into consideration when assigning the minimization of the mated station number as their objective function [46, 48, 51, 52].

In Özcan and Toklu (2009) [50] study, Eq. 2 becomes the mathematical formulation model for minimizing the mated station number besides being able to assist in minimizing the number of stations or workstations.

$$\text{Minimize} = \sum_{j\in J} \left(F_j + G_j\right) + \mathscr{E}.\sum_{j\in J} . \sum_{k=1,2} U_{jk} \tag{2}$$

where

$j$mated station

$k$side of the line; $k = \{$ 1, indicates a left 2, indicates a right

Jset of mated station; $J = \{1, 2,. .., j$

F_j1, if mated station j is utilized for both sides of the line; 0, otherwise

G_j1, if mated station j is utilized for only side of the line; 0, otherwise

$\mathscr{E}$a small positive value, $0 < \mathscr{E} \le 1/\ (2 * \text{nms} + 1)$

U_{jk}1, if stations j is utilized for only side of the line; 0, otherwise

The researches that choose the minimization of mated station number as their objective function have been reported in literature [4, 13, 16, 46, 48, 50–52]. The significant result has proven their accomplishment in the ALB studies.

4.3 Minimizing line length

Formerly, in the SALB production line, the longer and larger space are actually needed as only one side of assembly is used since 2S-ALB looks more reliable in dealing with this kind of problem. The 2S-ALB provides shorter length of line length than single-sided ALB [15]. This is due to the workstation dispensed on both sides of the assembly production systems, as shown in Fig. 2. A set of 2S-ALB assemblies will distribute

Table 2 Objective function for 2S-ALB problems

Author/s, year	Ref.	Objective Function											
		1	2	3	4	5	6	7	8	9	10	11	12
Yuan, Zhang et al. 2015	[4]	x	x										
Kucukkoc and Zhang 2015	[59]	x			x								
Kucukkoc and Zhang 2015	[60]	x											
Chiang, Urban et al. 2015	[15]	x		x									
Ağpak and Zolfaghari 2015	[53]	x		x									
Tuncel and Aydin 2014	[29]				x	x							x
Purnomo and Wee 2014	[30]										x		
Kucukkoc and Zhang 2014	[55]	x		x		x							
Kucukkoc and Zhang 2014	[54]	x		x									
Chutima and Naruemitwong 2014	[56]				x	x		x					
Purnomo, Wee et al. 2013	[12]				x								
Khorasanian, Hejazi et al. 2013	[48]	x	x			x							
Tapkan, Ozbakir et al. 2012	[61]	x											
Roshani, Fattahi et al. 2012	[49]											x	
Chutima and Chimklai 2012	[16]	x	x			x	x						
Ağpak, Yegül et al. 2012	[36]	x		x									
Taha, El-Kharbotly et al. 2011	[46]	x	x										
Özbakır and Tapkan 2011	[17]	x											
Xiaofeng, Erfei et al. 2010	[67]			x									
Özcan and Toklu 2010	[51]	x	x										
Özcan, Gökçen et al. 2010	[66]	x											
Özcan 2010	[13]	x	x										
Simaria and Vilarinho 2009	[14]	x											
Özcan and Toklu 2009	[50]	x	x										
Özcan and Toklu 2009	[52]	x	x										
Kim, Song et al. 2009	[45]				x								
Hu, Wu et al. 2008	[62]								x				
Baykasoglu and Dereli 2008	[47]	x					x						
Lee, Kim et al. 2001	[11]						x			x			
Kim, Kim et al. 2000	[44]	x											

Objective function: 1—min. number of workstation, 2—min. number of mated station, 3—min. line length, 4—min cycle time, 5—workload/task smoothness, 6—max. work relatedness, 7—min. production variance, 8—min. idle time, 9—max. slackness, 10—max. production rate, 11—cost oriented, 12—optm. specific constraints

all of the tasks in between a mated station; therefore, it allows the length of the lines to be shortened [54]. Besides, a shortened line may provide other additional benefits [15] like the following:

1. Reduces the cost of material handling
2. Able to reduce the equipment of tools and fixture by implementing tool sharing approach for opposite workstation

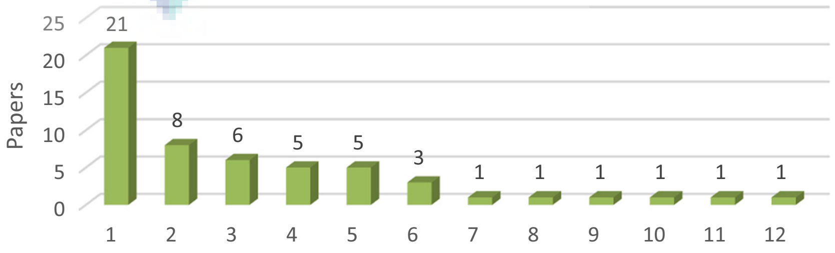


Fig. 6 Number of researches that used different objective functions in 2S-ALB

3. Reduce the overhead cost

   The formulation of line length minimization was presented in a study by Urban et al. (2015) [15]. They have formulated their objective function on minimizing the line length with w_1 and w_2 as the weight-associated parameters.

$$\min_{x_{jk}}\left\{w_1{}^*\max_k\left[\left(\overset{\cup}{_j}x^j{}_{jk}\right)*\left[\frac{k}{2}\right]\right]+w_2{}^*\sum_k\left(\overset{\cup}{_j}x^j{}_{jk}\right)\right\} \quad (3)$$

where

$j=1,2,\ldots,n$ tasks

$k=1,2,\ldots,m$ station

w_1, w_2 objective function weights for the line length and the number of stations, respectively

x_{jk}assignment variable, equal to one if task j is assigned to station k; equal to zero otherwise

In some studies, the minimization of line length was also called as the minimization of position number [36, 67]. The position number actually indicates the workstation in which it will reallocate and open in a row order [53]. Ağpak and Zolfaghari also succeeded in introducing a different evaluation in minimizing the line length. The formulation is as follows:

$$\text{Minimize } Z_2=\sum_{k=1}^{K}k.\ P_k \text{ or } Z_2=\sum_{k=1}^{K}P_k \quad (4)$$

where

$k$position, $k=1,2,\ldots,K$

P_k1, if any station at position k is open; 0, otherwise

The minimization of the line length could be suggested as the additional objective function in 2S-ALB or could be the secondary objective. In certain researches, the minimization of line length was performed after the minimization of workstation or number of mated station [53, 55]. Hence, a different idea and formulation has been built to represent 2S-ALB with success.

4.4 Minimizing cycle time

According to [59], the duration of cycle time is greatly related with workstation. In ALB, the minimization number of cycle time alternatively classifies performance as a type-2 ALB problem. The relation between the number of cycle time and the number of workstations in objective function has influenced various studies throughout their researches. Cycle time is commonly defined as the maximum time to complete any task allowed on each line of workstation. Such in 2S-ALB problem which definitely has two sides, either left or right of workstation, the cycle time will be strictly set as to not exceed the limit value of the processing task time. However, in many cases, the cycle time could not be filled by the task and it has created some gap on the workstation due to some restriction. Thus, the processing task time will not be equal to the assessed cycle time. In such cases, the gap associated with a void space is naturally called idle time.

The minimization of cycle time has been discussed in some previous researches. As in [45], they have set the minimization of cycle time as the single objective function. Eq. 6 is presented as the restriction for Eq. 5 to achieve the cycle time minimization.

$$\text{Minimize ct} \quad (5)$$

$$t_i^f \le ct \quad (6)$$

where

ct cycle time

t_i^f finish time of the task i

The summation of processing and idle time were actually performed as the general operation of calculating the number of cycle time [12], and it must be equal or smaller than the actual value [11, 36], which cannot exceed the designated cycle time. The minimization of cycle time is mentioned as equivalent to maximization of the assembly line efficiency (workstation efficiency) that reduces the idle time value. Workstation efficiency (WE) in Eq. 7 is defined by the total processing time of all tasks divided into time allocated in the workstation (cycle time) [12]. In measuring the workstation efficiency, the number of cycle time is also needed and becomes a factor for calculating the efficiency value.

$$\text{WE}=\frac{\sum_{i=1}^{n}t_i}{2.\text{m.}\ \text{CT}};\quad \text{i}\in\text{I} \quad (7)$$

where

st_isetup time

$t_i$processing time

Ia set of tasks assigned to the workstation

CTcycle time

In 2S-ALB, each line, left (L) and right (R), may have different numbers of cycle times [46]. Hence, it may have different throughput rates too. In addition, the sum of processing and idle time was actually performed as the general operation of calculating the number of cycle time [12] and it must be equal or smaller than the actual value [11, 36], which cannot exceed the designated cycle time. In measuring the workstation efficiency, the number of cycle time is also needed and becomes a factor for calculating the efficiency value [12].

The cycle time formulation determined by the researchers is normally connected with some restriction. In [29, 45, 56], they have prescribed the cycle time value into an amount which could not be greater. By this, the cycle time will not be exceeded and might be dropped. However, [59] has successfully applied a strict expression which combines two objective functions, that is, the cycle time and workstation minimization. The expression has also built together certain restriction constraints.

4.5 Workload/task smoothness

For any workstation on ALB problems, the assigned workload will not be same. Considering that, between the distributed tasks in industrial problems, the processing tasks are normally not equal. The assigned workload/task to the workstation initially is unbalanced. As in [29], the main goal is to improve the line balance implemented by the company for the given cycle time. Thus, considering the workload smoothness comes as another additional aim [14, 54]. Referring to [54, 55] that minimizes weighted idle times (WITs) also means ensuring a smooth workload among the workstations. Equation 8 below shows the expression of WIT.

$$WIT = \sum_{\varphi=1}^{\phi}\sum_{h=1}^{H}\sum_{k=1}^{K_h}\sum_{x\in\{0,1\}}\left(C-\sum_{j=1}^{M_h}\sum_{i=1}^{T_{hj}} op_{hj}\, pt_{hji} Y_{\text{hjikx}}^{\varphi}\right) \tag{8}$$

where

xSide of the line; $x=0$, which indicates left side of relevant line; 1, indicates right side of relevant line

φproduction cycle ($\varphi=1,\ldots,\phi$), where $\phi=LCM(S_1,\ldots,S_H)$

Ccommon cycle time for all lines

op_{hj}overall proportion of the demand of assembled product model

pt_{hji} processing time of task of t_{hji} model m_{hj} on line L_h

$Y_{\text{hjikx}}^{\varphi}$ 1, if task t_{hji} of model m_{hj} is assigned to station W_{hkx} on side x of line $L_h$in the production cycle φ; 0, otherwise

Smoothing the workload evenly is able to balance the workstation and the assembly line. In fact, it is successfully presented in [29], which balances using the Teaching–learning-based optimization (TLBO) algorithm. The smoothness index (Eq. 9) among the workstation is calculated within ranged value. The formulation is as follows:

$$C_b = \sum_{k=1}^{K}\left[\left(\frac{I_k}{T}\right)-(1/K)\right]^2 \tag{9}$$

where

$C_b$line smoothness index

Ktotal number of workstations utilized on the line

Ttotal idle time of all workstation

I_k idle time at workstation k

Once the calculation of smoothness index C_b is done, the line efficiency is shown to improve as well. Another equation contributed to the balance workload is illustrated in [16] whereby they assigned workload plus idle time for any workstation. Moreover, a uniform distribution across open workstation trusted has the same meaning as uniform idle time distribution. Therefore, the balance workload can be calculated from the following equation:

$$\text{Minimize } B_b = \frac{N_w}{N_w-1}\sum_{k=1}^{LL}\sum_{b=L}^{R}\left(\frac{S_{kb}}{WIT}-\frac{1}{N_w}\right)^2 \tag{10}$$

where

B_b workload balance between workstations

N_w the number of operators

S_{kb} the average idle time of workstation k on side b

WIT weighted idle time

The workload balance distribution among workstation is taken as counted measure since it is recommended in [14]. The recommendation is also highlighted in [16] and has been successfully presented in B_b formulation equation in terms of balancing the workload among the workstations.

4.6 Other objective functions

Another significant objective function prescribed by the researcher besides the above five examined earlier could have a big potential. They have noticed other critical objective function that could be used and applied in optimizing the assembly line, for instance, the minimization of the production variance [56] and the minimization of idle time [62]. Besides, the maximizing work relatedness [11, 16, 47], maximizing slackness [11], maximizing production rate [30], optimization of specific constraints [29], and cost oriented [49] can also bring great influence to other research.

5 Constraint

Normally, for every research on 2S-ALB, it will consist of feasible assignment with certain restrictions. In order to acquire more sensible and effective solution, the presented studies have considered the real non-obligatory relationships between tasks in assigning them to the workstations on the assembly lines [35, 48] such as the precedence relation constraint that indicates each operational process for every assigned task. It practically could not be considered because of the influences against all of the assembly operations. However, other constraints used in the 2S-ALB as the restriction will be discussed based on its popularity. Table 3 shows the constraints considered on the previous research of the 2S-ALB problems.

Figure 7 above has illustrated the frequency for each different type of constraints in 2S-ALB. The number under the graph represents the difference between optimization method types (see legend of Table 3). Zoning constraint leads the frequency graph (Fig. 7) by 16 counts followed by 13 cycle times and operation direction constraints with only 9 counts.

5.1 Zoning constraint

Large numbers of researches on ALB problems have been considered both by the academics and industry. While in 2S-ALB problems, various types of solution approach were suggested by the researcher in solving the faced problems. In order to reach the specified objective and succeed in the studies, most of the researchers strictly applied certain restrictions or constraints. As an example, Baykasoglu and Dereli (2008) studied the minimizing of the number of workstations in the 2S-ALB problems [47]. Some restrictions and constraints were built such as zoning. In some other researches, the zoning constraint was also known as the positional constraint.

Zoning constraint actually is a preference of task to be assigned on which workstation [16] on the assembly line. Respectively, the zoning constraints are divided into two, either positive or negative, zoning [30, 53, 60]. In general, positive zoning is a restriction for assigning more than one task into a workstation, while negative zoning strictly controls to not to be assigned with any set of tasks into the same workstation. Positive zoning is usually related to the common tools

Table 3 Constraints in 2S-ALB problems

Author/s, Year	Ref.	Constraint											
		1	2	3	4	5	6	7	8	9	10	11	12
Yuan, Zhang et al. 2015	[4]			x									
Kucukkoc and Zhang 2015	[59]				x		x			x			
Kucukkoc and Zhang 2015	[60]	x	x		x								
Chiang, Urban et al. 2015	[15]		x	x								x	
Ağpak and Zolfaghari 2015	[53]	x	x										
Tuncel and Aydin 2014	[29]	x				x	x	x					
Purnomo and Wee 2014	[30]	x											
Kucukkoc and Zhang 2014	[55]	x			x								
Kucukkoc and Zhang 2014	[54]	x		x	x				x	x			
Chutima and Naruemitwong 2014	[56]												x
Purnomo, Wee et al. 2013	[12]	x	x			x	x	x	x		x		
Khorasanian, Hejazi et al. 2013	[48]							x			x		
Tapkan, Ozbakir et al. 2012	[61]	x				x							
Roshani, Fattahi et al. 2012	[49]		x	x	x								
Chutima and Chimklai 2012	[16]	x											
Ağpak, Yegül et al. 2012	[36]	x	x										
Taha, El-Kharbotly et al. 2011	[46]			x									
Özbakır and Tapkan 2011	[17]	x											
Xiaofeng, Erfei et al. 2010	[67]			x									
Özcan and Toklu 2010	[51]		x				x						
Özcan, Gökçen et al. 2010	[66]		x										
Özcan 2010	[13]		x										
Simaria and Vilarinho 2009	[14]	x		x	x	x							
Özcan and Toklu 2009	[50]	x				x							
Özcan and Toklu 2009	[52]	x											
Kim, Song et al. 2009	[45]		x						x				
Hu, Wu et al. 2008	[62]		x	x									
Baykasoglu and Dereli 2008	[47]	x											
Lee, Kim et al. 2001	[11]		x	x									
Kim, Kim et al. 2000	[44]	x	x										

Constraints: 1—zoning, 2—cycle time, 3—operation direction, 4—capacity, 5—synchronous task, 6—workstation; 7—resource, 8—occurrence, 9—assignment, 10—distance, 11—completion probability, 12—sequence dependent task time

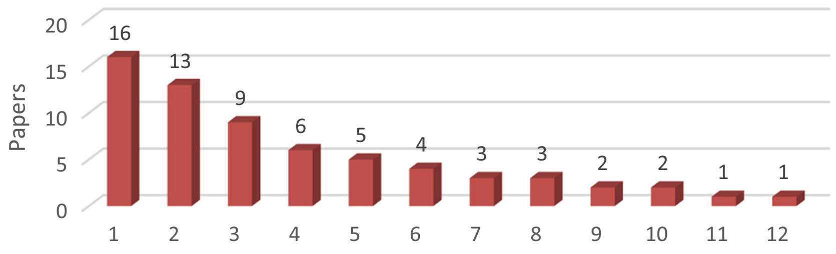


Fig. 7 Number of constraints used in the 2S-ALB reference paper

and fixture; therefore, the operational process could be assigned on the same workstation [29]. Meanwhile, negative zoning is something that is related to technology and equipment. Hence, it could not be assigned to the same workstation due to safety reasons or different required equipment [30].

There are two types of formulation in the zoning constraint, positive and negative. In studies by Simaria and Vilarinho (2009) [14] and Tapkan, Ozbakir et al. (2012) [61], they successfully performed both positive and negative zonings respectively for the same and different workstations. Equation 11 represents the positive zoning constraint where the set of tasks must be assigned at the same workstation, while Eq. 12 is the negative zoning constraint that the task must not be assigned to the same workstation.

$$\sum_K k(x_{ik1}+x_{ik2})-\sum_K k\left(x_{jk1}+x_{jk2}\right)=0 \qquad (i,j)\in ZP_{ij} \quad (11)$$

$$\sum_K k(x_{ik1}+x_{ik2})-\sum_K k\left(x_{jk1}+x_{jk2}\right)\neq 0 \qquad (i,j)\in ZN_{ij} \quad (12)$$

where

K	the set of workstations ($k=1,\ldots,I$)
ZP_{ij}	the set of pairs of tasks that must be assigned to the same workstation
ZN_{ij}	the set of pairs of tasks that cannot be assigned to the same workstation
x_{ikb}	$\{1$, if task i is assigned to workstation k at side b; 0, otherwise

However, different formulations of zoning constraint have been used by Wee et al. (2013) [12] and Yegül et al. (2012) [36] in their studies. Both formulations of Eqs. 13 and 14 were constructed based on the positive and negative zoning constraints (PZ and NZ) as well.

$$x_{gjk}-x_{ijk}=0 \qquad (g,i)\in \mathrm{PZ} \quad (13)$$

$$\mathrm{x}_{\mathrm{gjk}}+\mathrm{x}_{\mathrm{ijk}}\leq 1 \qquad (\mathrm{g},\mathrm{i})\in \mathrm{NZ} \quad (14)$$

where

x_{ijk}	a decision variable
(g,i)	distance between task g and i

The positive and negative formulations of the zoning constraints have been applied by numerous researchers throughout the significant ALB in different problem types [12, 14, 16, 17, 29, 30, 36, 44, 47, 50, 52–55, 60, 61].

5.2 Cycle time constraint

Another significant constraint to the line system is the duration of the entire processing time or usually called cycle time. These constraints were considered as one of the most important criteria to successfully balance the two-sided assemblies. Commonly, the cycle time is subjected for balancing purposes such as in simple ALB problems of type 1 and type 2. Both of these problems seriously considered cycle time as their objective.

Type 1:To minimize the number of workstations for a given cycle time

Type 2:To minimize the cycle time for a given number of workstations

From the two objectives above on the ALB problem, the cycle time could be best regarded as purely important either for retention (Type 1) or for reduction (Type 2).

Cycle time becomes important especially in measuring workstation efficiency (Eq. 7). For instance, it is required and becomes a factor for calculating the efficiency value [12]. Workstation efficiency (WE) in Eq. 7 is defined by the total processing time of all the tasks divided into the time allocated in the workstation (cycle time), while the response of changes on cycle time is formulated as the equation below [49]. Equation 15 is represented as the constraint that ensures the task will be finished before the cycle time ends.

$$st_i+t_i\leq CT, \qquad \mathrm{i}\forall\in \mathrm{I} \quad (15)$$

where

st_i	setup time
t_i	processing time

I	a set of tasks assigned to the workstation
CT	cycle time

The other cycle time formulation that takes idle time as the measure is shown in the equation below [12]. Equation 17 is the formulation of measuring the total of idle time in the workstation; meanwhile, the sum of processing and idle time is show in Eq. 16. The summation in Eq. 16 must be smaller than or equal to the cycle time while performing the cycle time restriction.

$$\sum_{i=1}^{n} t_i x_{ijk} + s_{jk} \leq CT \tag{16}$$

$$s_{jk} = \sum_{u=1}^{U} x_{ujk}\left(t_{u+1}^{s} - t_{u}^{f}\right) + \left(CT - t_{u}^{f}\right) \quad u \in Q_{jk} \tag{17}$$

where

t_i	processing time for task i
x_{ijk}	a decision variable
s_{jk}	total idle time at workstation j, side k
t_u^s	the starting time of task u
t_u^f	the finishing time of task u
CT	cycle time
Q_{jk}	a set of task that is assigned in workstation j side k

Recently, the cycle time constraint has become imperative to researchers in balancing purposes [11, 36, 44, 45, 51]. The high recommendations in every future study impacts the diverse utilization of the formulation. These are subjected to the different types of ALB problems that brought different ideas by different researchers in calculating the cycle time.

5.3 Operation direction constraint

A feasible balance line in 2S-ALB will be assigned with preferred sides of the line [15]. Therefore, allocating the task to the preferred workstation becomes crucial. This is the most challenging issue before completely running the assembly line throughout the desired task due to the 2S-ALB problems which are already categorized into three groups: left side (L), right side (R), and either sides (E) of the line [46, 49]. For this reason, the selection of the side was studied by the researcher. These are commonly called the operation direction constraint, and it should be fulfilled by the relations between every task. For example, the automotive assembly line which consists of the two-sided assembly operation. The left side usually will perform the task which prefers the left-hand handling, while the right side will perform the right-handed task. However, there are some tasks that do not have any preferred operation direction [49]. Hence, the proper selection of (left, right, or either) sides was greatly studied by researchers for optimization purposes.

The three equations below are the rule and formulation in selecting the preferred side (left, right, or either side) of the 2S-ALB problems [67]. The first equation, Eq. 18, will perform either side as the selection after calculating the total processing task t_i and t_j (cycle time). Then, Eqs. 19 and 20 will perform the left or right side of the selection after either side is filled (Eq. 18).

$$D(i) = E, \text{ if } \; t_i + t_j > C \;\; \forall j \in CTI_i \,, \tag{18}$$

then $t_i$is increased to C;

$$D(i) = L, \quad \text{if} \;\; t_i + t_j > C \;\; \forall j \in CTI_i, \mathrm{D}(j) \in \{L, E\}, \tag{19}$$

then t_i is increased to C;

$$D(i) = R, \quad \text{if} \;\; t_i + t_j > C \;\; \forall j \in CTI_i, \mathrm{D}(j) \in \{R, E\}, \tag{20}$$

then t_i is increased to C;

where

C	cycle time
i, j	task number
t_i	processing time of task i
t_j	processing time of task j
$D(i)$	operation direction of task i
$D(j)$	operation direction of task j

The operation direction constraint has been formulated with a definite purpose to allocate the preferred workstation whether left, right, or either side of the assembly line. Another research by Urban et al. (2015) also formulated its operation direction constraints in three main rules [15]. Each task will be assigned to only one station either left or right, starting with the left side. In Eq. 21, which is for the left operation side, the formulation is labeled with odd numbers $(1, 3, 5, \ldots, m-1)$ and even numbers $(2, 4, 6, \ldots, m)$ for the right-sided operation (Eq. 22). Therefore, Eq. 23 will be choosing either side of the assembly after both sides are filled.

$$\sum_{k \in \left\{1,3,5,\ldots,m-1\right.} x_{jk} = 1 \;\; \forall j \in L \tag{21}$$

$$\sum_{k \in \{2,4,6,\ldots,m\}} x_{jk} = 1 \;\; \forall j \in R \tag{22}$$

$$\sum_{k=1}^{m} x_{jk} = 1 \quad \forall j \in E \tag{23}$$

where

x_{jk}	assignment variable, equal to one if task j is assigned to station k, equal to zero otherwise

j 1, 2, …, *n* tasks
k 1, 2, …, *m* stations

The researchers who have taken the operation direction constraint into consideration significantly have been reported in the literature [4, 11, 15, 46, 49, 55, 62, 67]. Most of the studies that considered operation direction into their constraint from general ALB problem were due to the additional line to the assemblies.

5.4 Other constraints

Some other significant constraints that were used other than those reviewed above also possess their own abilities and advantages such as the capacity constraints that are commonly used in the line balancing problems, and they need to be satisfied. Commonly, the capacity constraint is developed by the total processing time of the assigned tasks to the workstation. If the next sequence task does not satisfy the restriction of the capacity constraint, a new workstation will be opened for the next assigned task [59, 60]. The capacity constraint also will ensure that the execution of each task is within the cycle time [55]. Other considered studies on capacity constraint are reviewed in other literature [14, 49, 54].

Besides that, the workstation constraints are also considered by some of the researchers in the 2S-ALB problems as the restriction. This constraint means, for each specific task, it will be assigned to a specific workstation. Therefore, the assigned task is strictly for a workstation where the task is really required [12, 86]. In a study by Tuncel and Aydin (2014), they have associated the workstation with particular equipment and material for the assembly operation. Thus, it also means a specific task could only be assigned to a certain and required workstation [29]. By this, workstation constraint seems absolutely essential in all ALB problem optimizations.

The assignment constraint to determine which task/assignment could be assigned in which location of the workstation was also studied. It also determines the duration of time in which the assignment must be executed at the same workstation when the current side task is lower than the opposite side task [54]. The assignment constraint could also ensure that each task will be assigned exactly once in completing the 2S-ALB operation [59]. The mated station in 2S-ALB always becomes another factor that may affect the completion probability. The completion probability constrain is constantly related with time. It is necessary for a mated station to complete the task given within the cycle time [15]. Therefore, the completion of time distribution must be determined explicitly in 2S-ALB for optimization purposes.

The synchronism of task in the single-sided ALB may not be very important, but in 2S-ALB, it is different. If a synchronization constrain is considered in 2S-ALB, the task will be divided to satisfy the synchronization constrain [29]. The identical task time for every mated station will perform the synchronism working experience, and it is only presented in 2S-ALB operating line. By doing this, the idle time of product processing will be minimized [12]. Other considered researches on the synchronization constraint are [14, 50, 61]. Another remarkable constraint in the 2S-ALB problems is the sequence-dependent task time. The existence of this type of constraint is to allocate each task into a workstation based on the preferred operational directions and precedence relationships [56]. The mated station factor could also affect the performance in the 2S-ALB; hence, this sequence-dependent task time constraint will allocate the task within the limited setup time.

The distance between tasks is formally able to become as a constraint, due to the important and affected factor regarding space and cycle time. Distance could be measured by the length of space, duration of time, and even workstation position, from the initial task to the next preferred task [12, 48]. Distance constraint will be set to the maximum or minimum with the aim to get prepared for the next task. For example, a painting process that needs a high duration of time to dry. Therefore, maximizing the distance will come into a way of preparing the paint before further tasks are done.

Different ALB operation requires different machines and tools. These will turn into an issue in ALB. Any equipment and tools for conducting a task are considered as a resource. Some researchers take resources as a restriction and consider them as constraints. Resource constraints might be in many forms such as space and operator [29]. An operator could be a resource to take the action to conduct the assembly operation. In Wee et al. (2013), the resource restriction is believed to be able to reveal the inadequate space for allocating the required machines on a workstation [12]. Meanwhile, several studies have highlighted the occurrence constraint in 2S-ALB. Occurrence constraint is to ensure that every task is assigned just only for one workstation [12, 45]. The researchers come with some formulation to control and act as desired for the optimization purposes.

Literally, different constraints actually come with a certain objective function. To meet the desired objective function, it needs a strict condition of constraint as a constraint will be set according to the considered objective function. For instance, the minimization number of workstations and the minimization of line length [61, 67] are also affected. The methods of choosing constraints are definitely different from one another. This is because the constraints will act as a strict condition to acquire the objective function value. Hence, the selection of a suitable constrain is seriously needed.

Besides, for more complex objective function, the constraint is significantly complex too. In [15, 53], the assessed objective function is presented in biobjective which aims at different targets. Although both studies searched for a similar objective function, the methods of choosing constrains are still quite different. Each study has provided their own way to specify how each selected constraint should behave in meeting the needs of objective function. Moreover, every constraint lives with a strength indicating how important it is to satisfy the considered objective function. A single constraint might still be a weak restriction to enforce the direction of the choosing objective function. Therefore, by picking a different constraint, it will ensure that the constraint becomes stronger and influences the other constraints to provide a good result in the objective function.

6 Discussion and research potential

Assembly line balancing (ALB) is a production planning problem concerned with allocating each task to the workstations on the assembly line. The purpose is to balance and optimize the assembly line to increase the productivity measure. Various methods have been established with different useful optimization techniques, and it has been expending a lot with new improvement and additional capability of solving methods. In identifying the research issue, four main specifications have been summarized and discussed accordingly. Considering the growing number of publication in solving ALB problems, this study was focusing only on the two-sided assembly line balancing (2S-ALB) problems because of its benefit and priority in solving large-manufactured products. The researches which deal with 2S-ALB problems on the algorithm optimization are classified as the NP-hard problems. However, as time goes by, more complicated types of problems appear. A different solving approach of different complex problems was presented. The complex combinatorial problems between two or three or even more problems were vigorously studied by the researcher to diversify the problem types and inspire other researchers for future studies. However, there are still a few unfilled potential and gaps through the ALB problem studies.

In ALB, there are two problem types which are simple and general that have been questioned by the researcher. Simple ALB is quite famous and extremely studied because of its ability to balance with only a single side of the operation line. Besides, it is able to optimize the operation line by simply allocating for the workstation without much hesitation. However, in manufacturing industries nowadays, a complex combination of problems is introduced. Because of the market and customer needs, the researcher has been extremely dedicated to the invention of a new ALB combinatorial problems, known as the general ALB, such as in Chutima and Chimklai (2012) and Zhang et al. (2015), who presented the combination of mixed-model and two-sided problems in ALB [4, 16]. Meanwhile, Kucukkoc and Zhang (2014) introduced the mixed-model parallel two-sided assembly line balancing problem [54, 55]. Figure 8 presents the problem combination graphs of 2S-ALB that have been considered by researchers since 2008. These absolutely change the assembly configuration but successfully provide more beneficial advantages. From these, it was noted that researchers were more interested in combining the 2S-ALB with the other ALB models. Hence, this paper focused on the study of 2S-ALB problems.

From most of the published works, the genetic algorithm (GA) has successfully dominated the 2S-ALB problem. As presented in Fig. 5, GA is found competitive in solving the different types of combinatorial ALB problems. The successfulness of the GA method in solving complex problems is well known, with the presented control parameter, such as population size and crossover probability. Besides GA, the popularity in solving the complex combinatorial 2S-ALB problem is followed by the ant colony optimization (ACO) algorithm method. The pheromone trail of the ant path by the ACO method became the idea to find the solution and solving method. The simulated annealing (SA) followed after both the above methods. Even so, numerous other metaheuristics such as bee algorithm (BA) and particle swarm optimization (PSO)

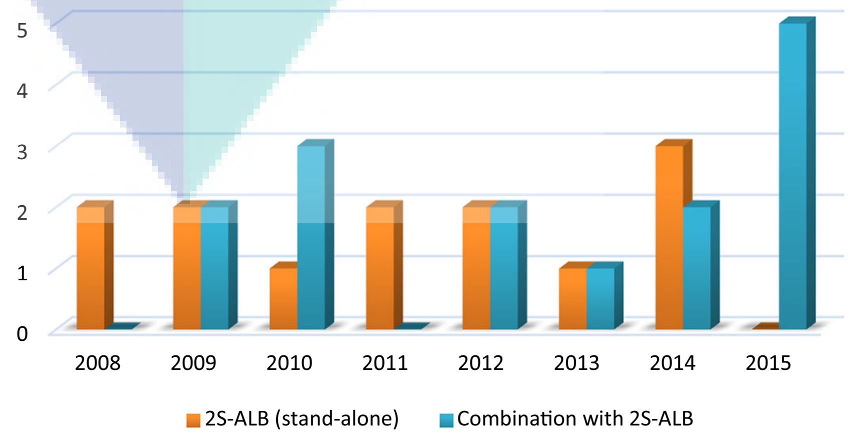


Fig. 8 2S-ALB problem combination trend by year

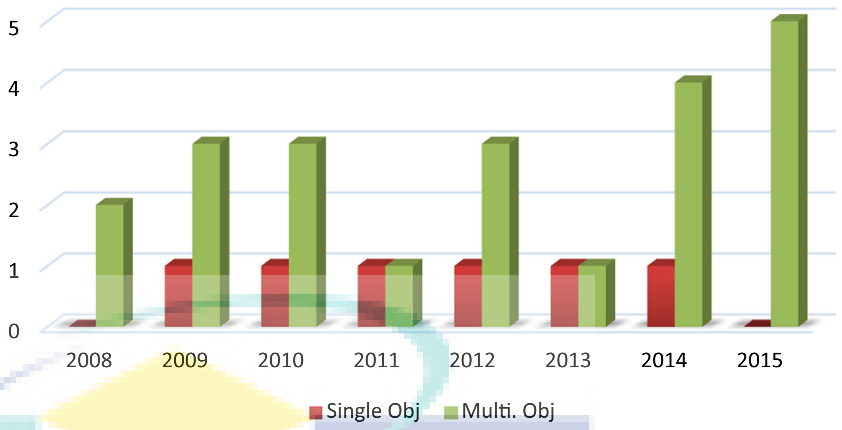


Fig. 9 The trend of 2S-ALB multiobjective by year

greatly had a big potential in solving the 2S-ALB combinatorial problems. From the assembly line balancing perspective, we also noted that the implementation of advanced computational method such as metaheuristic approaches are broadly studied compared with the heuristic, a problem-dependent solution approach.

Another issue that has been highlighted by researchers is the objective function consideration. It is a necessity for an ALB optimization research to indicate and point the objective function itself. This precisely will make sure the direction of the optimization research study. Most of the literature considers either maximizing or minimizing the appropriate task in fully optimizing the assembly line. Figure 6 illustrates the trend of objective function considered by researchers. Practically, in 2S-ALB, the majority considered the minimization number of workstation as the main objective. It is followed by the minimization of mated station number. The minimization number of workstation always looks significantly able to optimize the assembly line; however, it is not that simple. In other cases, the minimization of the number of workstations also might delay certain processes that would seriously reduce the productivity. By this, the automation and integration of assembly optimization has potential. Nowadays, a multiobjective function is widely studied, rather than a single-objective function. Figure 9 illustrates the trend of the multiobjectives that have been studied. This is due to the modern manufacturing system which always tries to fulfill the demands.

Besides that, in filling the needs of the objective function and plant configuration, some constraints will be counted. Therefore, normally, in every research of 2S-ALB, it will consist of certain restrictions. The considered constraint among the 2S-ALB researchers is shown in Fig. 7. The trend is to start with the zoning constraint, followed by cycle time and operation direction constraint. With this restriction on every constraint, it will effectively be presented by a certain rule or mathematical formulation. Because of the existence of a complex combination problem type with different designs of plant configuration, various constraints have been introduced recently. The presented constraints will act as the restriction to completely acquire the main objective of the research.

In dealing with various types of complex combination problems, various approaches of the optimization method and objective have been established. This is because the ALB problems are getting more complicated day after day. Hence, the growing of 2S-ALB researches in solving the manufacturing issues shall be supported by the manufacturing industry itself. Besides, the researcher might be able to manage and introduce more combinations or hybridized optimization methods on the ALB problems.

Since 2S-ALB is classified as an NP-hard combinatorial optimization problem, it consequently possesses a big concern to the researcher in finding the best solution. NP in fact stands for nondeterministic polynomial time, commonly informed as the hardest problem to solve and needs a large amount of time [44, 54]. Therefore, nowadays, the ALB problem has received widespread attention among researchers and industrial practitioners. At once, the hybridization technique between two algorithms successfully minimizes the computational time and makes the solving period faster. Most of the time, none of the researchers have guaranteed an optimal solution for the ALB problems, but they always relatively offer a good solution in reducing the computing period. Therefore, different solving approaches have been found, including metaheuristics, as the deficiency of the possible solving method should be considered for the enhancement as the potential research study.

7 Conclusions

Lately, the studies on ALB problems are growing very rapidly if compared to the previous decades, and most of the studies are focusing on the manufacturing industries. Industries, especially those manufacturing large-sized products, like cars, busses, or trucks, always need to keep improving, and the productivity will act as the measure. Thus, one of the ways to increase the company profit is by optimizing the assembly line. It is crucially important, especially in the assembly of large-sized products. In order to develop and optimize the ALB problems that are becoming complex day after day, different methods and solution approaches should be presented. There are still plenty of chances even in the algorithm development which is not being exposed yet.

In future research direction, various kinds of improvements that are precisely able to modify and develop the ALB problems are needed. The combination of certain general assembly lines are found to be competitive for being the best on making a closer model to the actual industrial circumstances. Besides that, the hybridization of an algorithm method also seems to be able to improve the assembly line problem. This method should also be implemented in 2S-ALB problems which formerly have been applied to the simple ALB problem. The potential of other heuristic or metaheuristic solving approaches also require a lot of attention. This can provide more alternative ways and methods in solving the ALB problems and thus able to hybridize and compare the effectiveness of the new algorithm method. Another direction should be focused to solve and optimize the customized problem/combination problem because the assembly line/production becomes more complex. Besides, the importance of automation is seen to be crucial. Further study related to ALB and automation is believed to be able to help to improve and provide more confidence to the industrial practitioner.

Acknowledgments The authors would like to acknowledge the Ministry of Higher Education, Malaysia and Universiti Malaysia Pahang for supporting this research under the FRGS grant RDU140103.

References

1. Becker C, Scholl A (2006) A survey on problems and methods in generalized assembly line balancing. Eur J Oper Res 168(3):694–715
2. Lam NT et al. (2016) Lean line balancing for an electronics assembly line. Procedia CIRP 40:437–442
3. Caggiano A, Marzano A, Teti R (2016) Resource efficient configuration of an aircraft assembly line. Procedia CIRP 41:236–241
4. Yuan B et al. (2015) An effective hybrid honey bee mating optimization algorithm for balancing mixed-model two-sided assembly lines. Comput Oper Res 53:32–41
5. Boysen N, Fliedner M, Scholl A (2007) A classification of assembly line balancing problems. Eur J Oper Res 183(2):674–693
6. Rashid MFF, Hutabarat W, Tiwari A (2011) A review on assembly sequence planning and assembly line balancing optimisation using soft computing approaches. Int J Adv Manuf Technol 59(1):335–349
7. Scholl A, Becker C (2006) State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. Eur J Oper Res 168(3):666–693
8. Erel E, Sarin SC (1998) A survey of the assembly line balancing procedures. Prod Plan Control 9(5):414–434
9. Sivasankaran P, Shahabudeen P (2014) Literature review of assembly line balancing problems. Int J Adv Manuf Technol 73(9):1665–1694
10. Kriengkorakot N, Pianthong N (2007) The assembly line balancing problem. KKU Eng J 34(2):133–140
11. Lee TO, Kim Y, Kim YK (2001) Two-sided assembly line balancing to maximize work relatedness and slackness. Comput Ind Eng 40(3):273–292
12. Purnomo HD, Wee H-M, Rau H (2013) Two-sided assembly lines balancing with assignment restrictions. Math Comput Model 57(1–2):189–199
13. Özcan U (2010) Balancing stochastic two-sided assembly lines: a chance-constrained, piecewise-linear, mixed integer program and a simulated annealing algorithm. Eur J Oper Res 205(1):81–97
14. Simaria AS, Vilarinho PM (2009) 2-ANTBAL: an ant colony optimisation algorithm for balancing two-sided assembly lines. Comput Ind Eng 56(2):489–506
15. Chiang W-C, Urban TL, Luo C (2015) Balancing stochastic two-sided assembly lines. Int J Prod Res:1–19
16. Chutima P, Chimklai P (2012) Multi-objective two-sided mixed-model assembly line balancing using particle swarm optimisation with negative knowledge. Comput Ind Eng 62(1):39–55
17. Özbakır L, Tapkan P (2011) Bee colony intelligence in zone constrained two-sided assembly line balancing problem. Expert Syst Appl 38(9):11947–11957
18. Boysen N, Fliedner M, Scholl A (2009) Sequencing mixed-model assembly lines: survey, classification and model critique. Eur J Oper Res 192(2):349–373
19. Sotskov YN et al. (2015) Enumerations and stability analysis of feasible and optimal line balances for simple assembly lines. Comput Ind Eng 90:241–258
20. Baykasoglu A (2006) Multi-rule multi-objective simulated annealing algorithm for straight and U type assembly line balancing problems. J Intell Manuf 17(2):217–232
21. Sungur B, Yavuz Y (2015) Assembly line balancing with hierarchical worker assignment. J Manuf Syst 37(Part 1):290–298
22. Esmaeilbeigi R, Naderi B, Charkhgard P (2015) The type E simple assembly line balancing problem: a mixed integer linear programming formulation. Comput Oper Res 64:168–177
23. García-Villoria A, Pastor R (2013) Erratum to "a solution procedure for type E simple assembly line balancing problem". Comput Ind Eng 66(1):201–202
24. Zacharia PT, Nearchou AC (2013) A meta-heuristic algorithm for the fuzzy assembly line balancing type-E problem. Comput Oper Res 40(12):3033–3044
25. Wei N-C, Chao IM (2011) A solution procedure for type E simple assembly line balancing problem. Comput Ind Eng 61(3):824–830
26. Micieta, B. and V. Stollomann, (2011) Assembly Line Balancing. DAAAM International p 257–264
27. Pearce, B.W. (2015) A Study on General Line Balancing Modeling Method and Techniques. Clemson University p 239
28. Tapkan P, Özbakır L, Baykasoğlu A (2016) Bee algorithms for parallel two-sided assembly line balancing problem with walking times. Appl Soft Comput 39:275–291
29. Tuncel G, Aydin D (2014) Two-sided assembly line balancing using teaching–learning based optimization algorithm. Comput Ind Eng 74:291–299
30. Purnomo HD, Wee H-M (2014) Maximizing production rate and workload balancing in a two-sided assembly line using harmony search. Comput Ind Eng 76:222–230
31. Sepahi, A. and S.G.J. Naini, Two sided assembly line balancing problem with parallel performing properties. Applied Mathematical Modelling, 2016.
32. Simaria AS, Vilarinho PM (2004) A genetic algorithm based approach to the mixed-model assembly line balancing problem of type II. Comput Ind Eng 47(4):391–407
33. Buyukozkan K et al. (2016) Lexicographic bottleneck mixed-model assembly line balancing problem: artificial bee colony and

tabu search approaches with optimised parameters. Expert Syst Appl 50:151–166
34. Vilarinho PM, Simaria AS (2002) A two-stage heuristic method for balancing mixed-model assembly lines with parallel workstations. Int J Prod Res 40(6):1405–1420
35. Merengo C, Nava F, Pozzetti A (1999) Balancing and sequencing manual mixed-model assembly lines. Int J Prod Res 37(12):2835–2860
36. Ağpak K, Yegül MF, Gökçen H (2012) Two-sided U-type assembly line balancing problem. Int J Prod Res 50(18):5035–5047
37. Kucukkoc I, Zhang DZ (2015) Balancing of parallel U-shaped assembly lines. Comput Oper Res 64:233–244
38. Alavidoost MH, Babazadeh H, Sayyari ST (2016) An interactive fuzzy programming approach for bi-objective straight and U-shaped assembly line balancing problem. Appl Soft Comput 40: 221–235
39. Ogan D, Azizoglu M (2015) A branch and bound method for the line balancing problem in U-shaped assembly lines with equipment requirements. J Manuf Syst 36:46–54
40. Hazır Ö, Dolgui A (2015) A decomposition based solution algorithm for U-type assembly line balancing with interval data. Comput Oper Res 59:126–131
41. Mosadegh H, Zandieh M, Ghomi SMTF (2012) Simultaneous solving of balancing and sequencing problems with station-dependent assembly times for mixed-model assembly lines. Appl Soft Comput 12(4):1359–1370
42. Manavizadeh N et al. (2013) A simulated annealing algorithm for a mixed model assembly U-line balancing type-I problem considering human efficiency and just-in-time approach. Comput Ind Eng 64(2):669–685
43. Bartholdi JJ (1993) Balancing two-sided assembly lines: a case study. Int J Prod Res 31(10):2447–2461
44. Kim YK, Kim Y, Kim YJ (2000) Two-sided assembly line balancing: a genetic algorithm approach. Prod Plan Control 11(1): 44–53
45. Kim YK, Song WS, Kim JH (2009) A mathematical model and a genetic algorithm for two-sided assembly line balancing. Comput Oper Res 36(3):853–865
46. Taha RB et al. (2011) A genetic algorithm for solving two-sided assembly line balancing problems. Ain Shams Eng J 2(3–4):227–240
47. Baykasoglu A, Dereli T (2008) Two-sided assembly line balancing using an ant-colony-based heuristic. Int J Adv Manuf Technol 36(5–6):582–588
48. Khorasanian D, Hejazi SR, Moslehi G (2013) Two-sided assembly line balancing considering the relationships between tasks. Comput Ind Eng 66(4):1096–1105
49. Roshani A et al. (2012) Cost-oriented two-sided assembly line balancing problem: a simulated annealing approach. Int J Comput Integr Manuf 25(8):689–715
50. Özcan U, Toklu B (2009) Balancing of mixed-model two-sided assembly lines. Comput Ind Eng 57(1):217–227
51. Özcan U, Toklu B (2010) Balancing two-sided assembly lines with sequence-dependent setup times. Int J Prod Res 48(18):5363–5383
52. Özcan U, Toklu B (2009) Multiple-criteria decision-making in two-sided assembly line balancing: a goal programming and a fuzzy goal programming models. Comput Oper Res 36(6):1955–1965
53. Ağpak K, Zolfaghari S (2015) Mathematical models for parallel two-sided assembly line balancing problems and extensions. Int J Prod Res 53(4):1242–1254
54. Kucukkoc I, Zhang DZ (2014) Simultaneous balancing and sequencing of mixed-model parallel two-sided assembly lines. Int J Prod Res 52(12):3665–3687
55. Kucukkoc I, Zhang DZ (2014) Mathematical model and agent based solution approach for the simultaneous balancing and sequencing of mixed-model parallel two-sided assembly lines. Int J Prod Econ 158:314–333
56. Chutima P, Naruemitwong W (2014) A Pareto biogeography-based optimisation for multi-objective two-sided assembly line sequencing problems with a learning effect. Comput Ind Eng 69:89–104
57. Yang X-S (2010) Engineering optimization an introduction with metaheuristic applications. John Wiley & Sons, Inc, Hoboken
58. Kucukkoc I, Karaoglan AD, Yaman R (2013) Using response surface design to determine the optimal parameters of genetic algorithm and a case study. Int J Prod Res 51(17):5039–5054
59. Kucukkoc I, Zhang DZ (2015) Type-E parallel two-sided assembly line balancing problem: mathematical model and ant colony optimisation based approach with optimised parameters. Comput Ind Eng 84:56–69
60. Kucukkoc I, Zhang DZ (2015) A mathematical model and genetic algorithm-based approach for parallel two-sided assembly line balancing problem. Prod Plan Control 26(11):874–894
61. Tapkan P, Ozbakir L, Baykasoglu A (2012) Modeling and solving constrained two-sided assembly line balancing problem via bee algorithms. Appl Soft Comput 12(11):3343–3355
62. Hu X, Wu E, Jin Y (2008) A station-oriented enumerative algorithm for two-sided assembly line balancing. Eur J Oper Res 186(1):435–440
63. Ding F-Y, Zhu J, Sun H (2006) Comparing two weighted approaches for sequencing mixed-model assembly lines with multiple objectives. Int J Prod Econ 102(1):108–131
64. Gökçen H, Ağpak K, Benzer R (2006) Balancing of parallel assembly lines. Int J Prod Econ 103(2):600–609
65. Çerçioğlu H et al. (2009) A simulated annealing approach for parallel assembly line balancing problem. J Fac Eng Archit Gazi Univ 24(2):331–341
66. Özcan U, Gökçen H, Toklu B (2010) Balancing parallel two-sided assembly lines. Int J Prod Res 48(16):4767–4784
67. Xiaofeng H et al. (2010) A branch-and-bound algorithm to minimize the line length of a two-sided assembly line. Eur J Oper Res 206(3):703–707
68. Li, S.-x. and H.-b. Shan. GSSA and ACO for assembly sequence planning: a comparative study. in Automation and Logistics, 2008. ICAL 2008. IEEE International Conference on. 2008.
69. Shan, H., et al. (2006) Genetic simulated annealing algorithm-based assembly sequence planning. in Technology and Innovation Conference. ITIC 2006. International 2006.
70. Moon I, Logendran R, Lee J (2009) Integrated assembly line balancing with resource restrictions. Int J Prod Res 47(19):5525–5541
71. Tasan SO, Tunali S (2007) A review of the current applications of genetic algorithms in assembly line balancing. J Intell Manuf 19(1): 49–69
72. Dorigo, M. and C. Blum, Ant colony optimization theory: a survey. Theor Comput Sci, 2005. 344(2–3): p. 243–278.
73. Zhang, Z.-q., et al. (2007) Ant algorithm with summation rules for assembly line balancing problem. In Management Science and Engineering. ICMSE 2007. International Conference on. 200
74. Spinellis DD, Papadopoulos CT (2000) A simulated annealing approach for buffer allocation in reliable production lines. Ann Oper Res 93(1):373–384
75. Chen S-F, Liu Y-J (2001) An adaptive genetic assembly-sequence planner. Int J Comput Integr Manuf 14(5):489–500
76. Sabuncuoglu I, Erel E, Tanyer M (2000) Assembly line balancing using genetic algorithms. J Intell Manuf 11(3):295–310
77. Zhao YZ, de Souza R (2000) Genetic production line-balancing for the hard disk drive industry. Int J Adv Manuf Technol 16(4):297–302
78. Gonçalves JF, de Almeida JR (2002) A hybrid genetic algorithm for assembly line balancing. J Heuristics 8(6):629–642
79. Wang JF, Liu JH, Zhong YF (2005) A novel ant colony algorithm for assembly sequence planning. Int J Adv Manuf Technol 25(11): 1137–1143

80. Shuang B, Chen J, Li Z (2008) Microrobot based micro-assembly sequence planning with hybrid ant colony algorithm. Int J Adv Manuf Technol 38(11):1227–1235
81. Baykasoglu, A., et al. (2009) Balancing parallel assembly lines via ant colony optimization. In Computers and Industrial Engineering. CIE 2009. International Conference on. 2009
82. Aydin ME, Fogarty TC (2004) A distributed evolutionary simulated annealing algorithm for combinatorial optimisation problems. J Heuristics 10(3):269–292
83. Mahfoud SW, Goldberg DE (1995) Parallel recombinative simulated annealing: a genetic algorithm. Parallel Comput 21(1):1–28
84. Kennedy, J. and R. Eberhart. (1995) Particle swarm optimization. in neural networks. Proceedings, IEEE International Conference on. 1995
85. Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11(4):341–359
86. Lapierre SD, Ruiz AB (2004) Balancing assembly lines: an industrial case study. J Oper Res Soc 55(6):589–597

CHAPTER 3

PROBLEM MODELLING

This chapter present the problem modelling and formulation for ALB problems. This consists of modelling for ALBRC. The detail of the problem modelling were presented in two papers as follow:

1. Razali, M.M., Ab. Rashid, M.F.F. and Abdullah Make, M.R. (2016), "Mathematical Modelling of Mixed-Model Assembly Line Balancing Problem with Resources Constraints", IOP Conference Series: Materials Science and Engineering, Vol. 160, No. 1, pp. 1-10.
2. Abdullah Make, M.R., Ab. Rashid, M.F.F. and Razali, M.M. (2016), "Modelling of Two-sided Assembly Line Balancing Problem with Resource Constraints", IOP Conference Series: Materials Science and Engineering, Vol. 160, No. 1, pp. 1-9.

Mathematical Modelling of Mixed-Model Assembly Line Balancing Problem with Resources Constraints

Mathematical Modelling of Mixed-Model Assembly Line Balancing Problem with Resources Constraints

Muhamad Magffierah Razali, Mohd Fadzil Faisae Ab.Rashid, Muhammad Razif Abdullah Make

Faculty of Mechanical Engineering, Universiti Malaysia Pahang, 26600 Pekan, Pahang, Malaysia

E-mail: magffierah_razali@yahoo.com, ffaisae@ump.edu.my, razifabdullahmake@gmail.com

Abstract. Modern manufacturing industries nowadays encounter with the challenges to provide a product variety in their production at a cheaper cost. This situation requires for a system that flexible with cost competent such as Mixed-Model Assembly Line. This paper developed a mathematical model for Mixed-Model Assembly Line Balancing Problem (MMALBP). In addition to the existing works that consider minimize cycle time, workstation and product rate variation, this paper also consider the resources constraint in the problem modelling. Based on the finding, the modelling results achieved by using computational method were in line with the manual calculation for the evaluated objective functions. Hence, it provided an evidence to verify the developed mathematical model for MMALBP. Implications of the results and future research directions were also presented in this paper.

1. Introduction

Over the year, assembly line balancing (ALB) problem has earned a lot of attention. The purpose of ALB is to distribute different tasks to the operators for a various workstation on the line in a way where the tasks do not violate any of the precedence restrictions and some measurements of effectiveness are being optimized [1]. ALB has been evaluated widely in the relevant literature by Becker and Scholl [2, 3]. Mixed-Model ALB is categorized under general assembly line balancing which produce several models having similar characteristics on a single assembly line [4]. Usually in a mixed-model assembly line, the models being assembled have differences in the set of tasks associated with each model, the processing times, precedence relations, and amount of production. With all due respect to all of these conditions, Mixed-Model Assembly Line Balancing Problem (MMALBP) has been categorized as an NP-hard combinatorial optimization problem as well as CPU time-consuming [5].

Mixed-model assembly lines solution procedures in the relevant literature was proposed by Thomopoulos [6]. Thomopoulos classify the procedures into three categories: meta-heuristics and heuristics, hybrid, and mathematical model solutions. In general, there are MMALBP-I and MMALBP-II, which aim to minimize number of workstations for a given amount of cycle time and to minimize cycle time for a given number of workstations respectively [7]. Due to the current scenario of global market, companies changed single model lines into mixed-model lines in order to provide diversity and meet customized customer needs on time in a perceptive manner [8]. In addition to that advantage, mixed-model lines do not require a new set-up process between model changes, provide a continuous flow of materials, reduce the inventory levels of final items, and very flexible with respect to model changes [9].

In MMALBP context, different problem model has been proposed with various objective functions. A mixed integer linear programming model has been proposed by Akpinar and Baykasoglu with the objectives to minimize the total number of workstations subjected to assignment, capacity and zoning constraints [10]. Two objectives being handled by Yagmahan that utilized an approach of multi-objective ant colony optimization which is maximizing the smoothness index between stations and minimizing the number of workstations by considering the precedence constraints [11]. Tiacci considers buffer allocation problem and assembly line balancing problem simultaneously by using genetic algorithm approach. The objective function called normalize design cost (NDC) is introduced and subjected to precedence restriction [12].

The mixed-model parallel two-sided assembly considered by Kucukkoc consist an objective function of minimize total number of utilised workstations, as well as to ensure a smooth workload (WS) among the stations from cycle to cycle and the constraints are model and task occurrence constraint, task assignment, and operation direction constraint [13]. Ant colony optimization algorithm for balancing mixed-model assembly lines (ANTBAL) is used by Vilarinho and Simaria together with an objective function to balance the workloads within each workstation and maximize weighted line efficiency by considering zoning and capacity constraints [14]. Hamzadayi aims at minimizing the number of the stations required on the line, smoothing the workload of stations between cycles as well as smoothing the workload of all stations within any cycle by take into account the restriction of parallel and zoning constraints [15].

Based on the literature review, there are some published works considering the resource constraint in their work but limited to a specific constraint. This paper will focus to propose a mathematical model for MMALBP-II to minimize general resource constraints along with the cycle time and product rate variation. MMALBP-II has been chosen because one of the objective functions in this paper is to minimize the cycle time for a given number of workstations and categorized under MMALBP-II problems.

2. Problem description and formulation

In this paper, the MMALB problem is formulated with the aims to minimize total cycle time, resources and product rate variation (PRV) by considering the resources constraint. In order to explain the problem formulation, an example of assembly problem consists of six tasks for three different models as shown in Fig.1. The general assumptions of the problem are as follows:

- A number of J models will be assembled on a mixed model assembly line.
- Assembly tasks for different models are almost similar, so we can suppose a combine precedence diagram for all of the models. Now, if some models do not use some tasks in their assembly process, the relevant task time will be 0.
- Operating or processing times related to a task is the same for each different models.
- Each task type is assigned to only one station regardless of models. Hence, tasks are not assigned to different stations for different models.
- Each operator works only on a single station, only one operator carries out the assembly tasks on each station and the tasks are undividable.
- Assembly models will be assembling with the same rate and consecutively.

By referring the above mentioned assumptions, the parameters and indices of the model will be as:

Notation	**Definition**
S	number of workstations(fixed) s=1,2…, S
J	number of product models to be assembled j=1,2…, J
Ne	number of task e=1,2…, Ne
pre_i	predecessor for task i based on precedence diagram
t_i	execution time for task i
D_T	total quantity of units or total demand
d_j	demand for product j, j = 1, 2, . . ., a
$X_{i,k}$	total quantity of product/produced over stages 1 to k, k = 1, 2, . . ., Dt
$maxR$	maximum resources r=1,2,…,$maxR$
C_T	cycle time
Tej	shift task model time
Te	shift task time
te	task time
Nj	demand schedule for each model
U	production rates variation of production sequence

Decision variables

Uej	1 if task, e is used on model j ; 0,otherwise
X_{es}	1 if task, e is assigned to workstation s; 0, otherwise
Y_{rs}	1 if resource, r is used in workstation s; 0,otheriwse

2.1. Mathematical modelling

A sample problem used in this paper for a modelling purpose is adapted based on Thomopoulos [16]. The assembly data being considered are consist of six tasks for three different models as shown in Fig.1 and indirectly depict the build relationship among all the tasks. The models consist of six tasks (denoted by 1 to 6). The task is represented by circles and the connecting arrows identify the immediate predecessor tasks. It shows which tasks can begin without any predecessor tasks, and which tasks have predecessor tasks. The sequence of tasks moves from left to right.

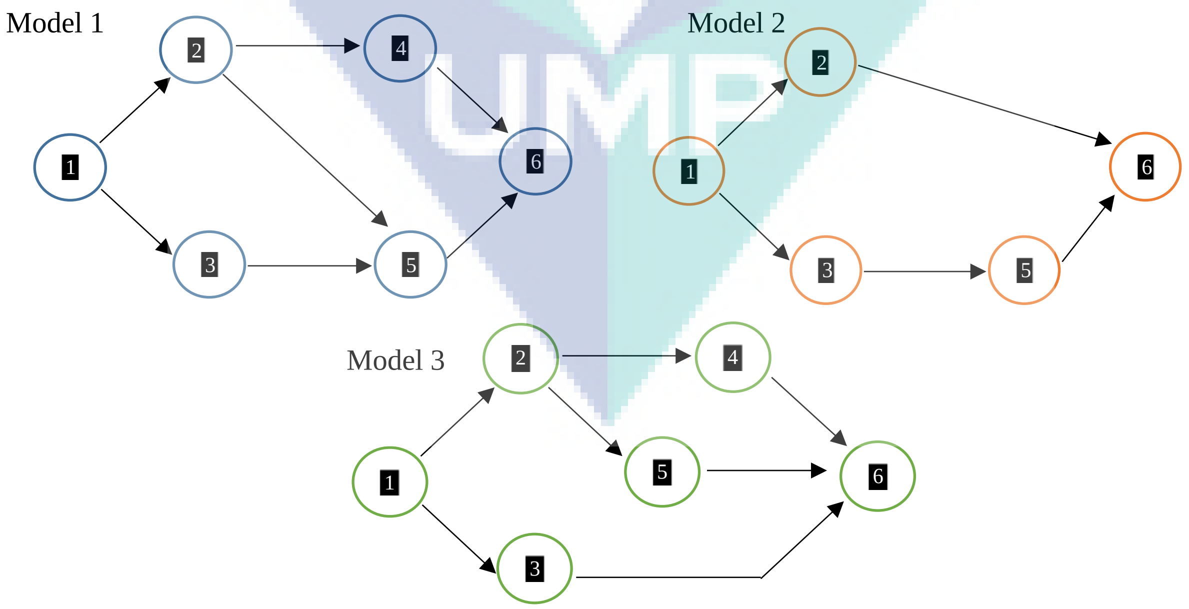


Figure 1. Precedence diagram for model 1, 2 and 3

Model 1: Task 1 on the left-hand side of the diagram and have no predecessor tasks. Task 2 and 3 cannot begin until Task 1 is completed. Also for instance, Task 5 cannot begin until Task 2 and 3 are completed. Lastly, Task 6 cannot begin until Task 4 and 5 are completed.
Model 2: Task 1 on the left-hand side of the diagram and have no predecessor tasks. Task 2 and 3 cannot begin until Task 1 is completed. Task 5 cannot begin until Task 3 is completed. Lastly, Task 6 cannot begin until Task 2 and 5 are completed.
Model 3: Task 1 on the left-hand side of the diagram and have no predecessor tasks. Task 2 and 3 cannot begin until Task 1 is completed. Task 4 and 5 cannot begin until Task 2 is completed. Lastly, Task 6 cannot begin until Task 3, 4 and 5 are completed.

The other important data are needed such as assembly time, type of resources used on the line, and model demands. Each model has their own task time as well as precedence relation but with the help of the anticipated model mix, a joint precedence graph is deduced from Figure 1.

2.2. Objectives function
The first objective function considered is to minimize cycle time. In our work, the cycle time is based on the product demand. Second objective function is minimizing total number of resource. The usage of resources such as tool, worker, and workstation is inevitable in an assembly line. However, based on the literature review less attention has been paid to minimize total number of resources even though the effect it can give to the operation cost is significant. Third objective function is product rate variation (PRV) which is common problem in the MMALB based on [17] . The problem formulation for this problem is presented as follows:

$$f1 = min \sum_{e=1}^{N_e} \sum_{j=1}^{J} C_T \tag{1}$$

$$f2 = min \sum_{s=1}^{S} \sum_{r=1}^{max_R} Y_{rs} \tag{2}$$

$$f3 = \min \sum_{k=1}^{D_T} \sum_{j=1}^{J} \left(x_{j,k} - k \times \frac{d_j}{D_T} \right)^2 \tag{3}$$

Objective function 1, *f1* in equation (1) is aim to minimize cycle time for a given number of workstation. Equation (2) targeted to minimize resources used on assembly line and equation (3) aim to minimize product rate variation (PRV). These objective functions are subjected to these constraints:

2.3. Constraints

$$\sum_{s=1}^{S} X_{es} = 1, \qquad e = 1, \dots, n \tag{4}$$

$$\sum_{s=1}^{S} X_{as} - \sum_{s=1}^{S} X_{bs} \leq 0, for\ \forall (a,b) \in pre_i \tag{5}$$

$$\sum_{i \in wk} t_i(X_{es}) \leq C, \qquad s = 1, \dots, S \tag{6}$$

Constraint (4) is to ensure that each task can only be assigned to one workstation [18]. Inequality (5) describes the precedence constraints among the tasks. It ensures that no successor of a task is assigned to an earlier station than that task. Constraint (6) ensures that the sum of task times assigned to each station does not exceed the cycle time. The maximum cycle time being considered here is stated as reference cycle time, RefCT which is expressed as:

$$Ref_{CT} = \frac{\sum shift\ task\ time, T_e}{no.\ of\ workstation, s}, \quad s = 1, \dots, S \tag{7}$$

Since this paper involve multi-objective optimization, the weighted sum approach is used as follows:

$$\sum_{i=1}^{M} w_i f_i(x) \quad ; w_1 f_1(x) + w_2 f_2(x) + \dots + \omega_n f_n(x) \tag{8}$$

In this case, the objective functions in equation 1, 2 and 3 needs to be normalized to ensure a consistent scaling to each of objective function. This is conducted by dividing the fitness value with the maximum value for each objective function. Thus, for future work the solution methods for solving this multi-objective problems are more simple and used a direct computation [19]. Equation (9) below represent the normalized fitness functions after using weighted sum approach.

$$F(X) = w_1 f_1'(x) + w_2 f_2'(x) + w_3 f_3{}'(x) \tag{9}$$

3. Numerical example

The generation of a joint precedence graph based on a combination for all three models are demonstrated in Figure 2 whereas Table 1 depicts the precedence matrix for this example. All known solution procedures for MMALB problem rely on the joint precedence graph which is, thus, indispensable for solving such problems. In short, this joint precedence diagram is like a blueprint on how to assemble the unit.

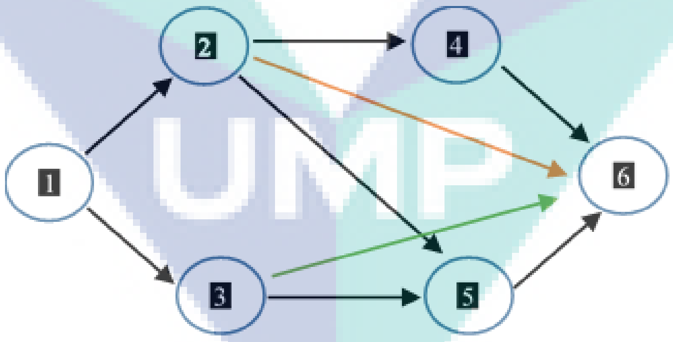


Figure 2. Joint precedence diagram

The example problem contain six tasks and three different model overall as referring to Table 1 and Table 2. Both shows the precedence table and model usage, *Uej* respectively. Model is represented by model 1, 2 and 3 and the demand for each model is 5, 3, and 2 respectively. For the model usage, if any of the task e is used on certain model, it is marked with 1, and 0 if otherwise.

Table 1. Task, *e*, task time, *te*, predecessor element, *p*

e	*te*	*p*
1	25	-
2	17	1
3	8	1
4	40	2
5	11	2,3
6	33	4,5

Table 2. Task, *e*, task time, *te*, model usage, U_{ej}

e	*te*		*Uej*	
		1	2	3
1	25	1	1	1
2	17	1	1	1
3	8	1	1	1
4	40	1	0	1
5	11	1	1	1
6	33	1	1	1

The modelling here also considers a resource that was used on the assembly lines. In this particular case, there will be three types of resources, R_T included tools, machines, and jigs. Table 3 summarizes the resources usage.

Table 3. Resources data tabulation

e	*te*			Resources, R_T		
1	25	T1	T2	J1	T3	-
2	17	T1	M2	J2	-	-
3	8	J1	-	-	-	-
4	40	T2	M1	T1	-	-
5	11	T3	J3	-	-	-
6	33	T3	M1	J2	M3	-

T1: Tool 1, **T2**: Tool 2, **T3**: Tool 3, **M1**: Machine 1, **M2**: Machine 2, **M3**: Machine 3, **J1**: Jig 1, **J2**: Jig 2, **J3**: Jig 3

Referring to mathematical model coded into MATLAB, all the selected sequence subjected to Ref_{CT} of 956.6667 minutes. Mean that, the specific tasks time that will be grouped into specific workstation must not exceed Ref_{CT} and only the last workstation can discard that rule since we will have the remaining task time to be converge at the last workstation which in this case workstation 3 (noted that maximum number of workstation is predetermined with total of three workstation).

In order to determine the performance for a particular assembly sequence, an evaluation needs to be conducted by comparing the objective functions. For assembly sequence [1 2 4 3 5 6], the procedure to calculate objective functions are presented in the following sections. For this particular assembly sequence, the computational test from MATLAB give a result of 1300 minutes for cycle time, 15 resources, and 2.9 rate of product variation. Based on the demand of 5, 3, and 2 for model 1, 2, and 3 respectively, the mathematical model construct here can be calculated manually in order to verify the results for each objective function with the result obtained from computational model.

3.1. Cycle time calculation

For the sequence of task stated above, the task time for each task is tabulated in Table 4 below and the task distribution for each workstation is described together.

Table 4. Task time for each sequence

Sequence of task	1	2	4	3	5	6
Model task time(minute)	625	425	520	200	275	825

- 625 **}WS 1**
- 425 + 520 = 945 **}WS 2**
- 200 + 275 + 825 = 1300 **}WS 3**

Based on the manual calculation above, the maximum cycle time for the selected sequence is 1300 minutes and we can assign the task to its own workstation subjected to Ref_{CT}. Task 1 is assigned to workstation 1, task 2 & 4 to workstation 2, and task 3, 5, & 6 to workstation 3 as shown in the Table 5 below.

Table 5. Workstation distribution

Sequence of task	1	2	4	3	5	6
Model task time(min)	625	425	520	200	275	825
Workstation time(min	625	945			1300	
Workstation, *s*	1	2			3	

3.2. Resources used on the line

Based on the assembly tasks assignment in section 3.1, the second objective function can be calculated. The manual calculation for the number of resources is illustrated in Table 6. The number of resources is determined by the different resource type in a workstation. For example, in workstation 2, since tasks 2 and 4 used similar T1 resource, the total resources in this workstation is equivalent to 5.

Table 6. Resources used by each workstation

Workstation	1		2		3	
Sequence of task	1	2	4	3	5	6
Resources type	T1	T1	T1	J1	T3	T3
	T2	M2	T2	-	J3	M1
	T3	J2	M1	-	-	M2
	J1	-	-	-	-	J2
$\boldsymbol{Y_{rs}}$	$\sum$= 4		$\sum$= 5		$\sum$= 6	

Therefore, by using the objective function definition for resource used on assembly line given in equation (2), the summation of resources used by each workstation are as followed:

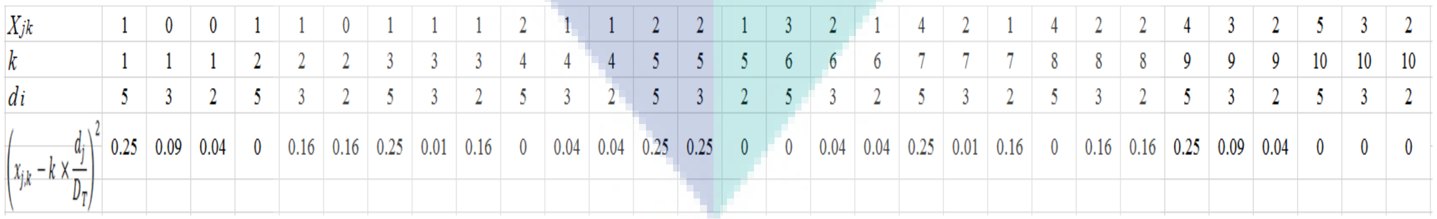
$$f2 = min \sum_{s=1}^{S} \sum_{r=1}^{max_R} Y_{rs}$$

= 4 + 5 + 6
= 15 resources

3.3. Product rate variation (PRV)
Product rate variation exist due to the nature of MMALBP itself which capable to assemble more than one product at a time and in this paper three different model being assemble on the same assembly line. The objectives of PRV problem is to achieve a stable production rate for each product. For instance, the target is to assemble 10 unit of particular product denoted by *k*, and we have a sequence of product, *j*.

Sequence of product, *j*	1	2	3	1	2	1	1	3	2	1

Based on the definition of objective function in (3), the value for each parameter have been calculated and summarized in Fig. 3.



X_{jk}	1	0	0	1	1	0	1	1	1	2	1	1	2	2	1
k	1	1	1	2	2	2	3	3	3	4	4	4	5	5	5
di	5	3	2	5	3	2	5	3	2	5	3	2	5	3	2
$\left(x_{j,k} - k \times \frac{d_j}{D_T}\right)^2$	0.25	0.09	0.04	0	0.16	0.16	0.25	0.01	0.16	0	0.04	0.04	0.25	0.25	0

X_{jk}	3	2	1	4	2	1	4	2	2	4	3	2	5	3	2
k	6	6	6	7	7	7	8	8	8	9	9	9	10	10	10
di	5	3	2	5	3	2	5	3	2	5	3	2	5	3	2
$\left(x_{j,k} - k \times \frac{d_j}{D_T}\right)^2$	0	0.04	0.04	0.25	0.01	0.16	0	0.16	0.16	0.25	0.09	0.04	0	0	0

Figure 3. Product rate variation summarization

Hence, the summation of $\left(x_{j,k} - k \times \frac{d_j}{D_T}\right)^2$ in the table is the PRV value that needed which resulted to 2.9 for the selected sequence of product, *j*.

4. Conclusion and future work
This paper proposed a mathematical model on MMALBP which consider minimize cycle time, minimize resources used on assembly line and minimize product rate variation (PRV) as the objectives. The computational model was developed using MATLAB software. In order to verify the output from the computational model, manual calculations had been carried out to compare the output.

From the output of both methods, we can conclude that the objective function in term of its mathematical expression is valid to be used since the acquired results for each objective function is the same either for computerized method or manual computation. Future work will consist of an optimization procedure for the proposed models earlier by using an artificial intelligence approach such as genetic algorithm, particle swarm optimization and simulated annealing.

Acknowledgement
The research described in this paper was performed under the research grant RDU140103 from Ministry of Higher Education Malaysia.

References
[1] A. K. Elia, "Minimizing Number of Stations and Cycle Time for Mixed Model Assembly Line," vol. 3, no. 8, pp. 1359–1364, 2014.
[2] C. Becker and A. Scholl, "A survey on problems and methods in generalized assembly line balancing," *Eur. J. Oper. Res.*, vol. 168, no. 3, pp. 694–715, 2006.
[3] A. Scholl and C. Becker, "State-of-the-art exact and heuristic solution procedures for simple assembly line balancing," *Eur. J. Oper. Res.*, vol. 168, no. 3, pp. 666–693, Feb. 2006.
[4] N. Boysen, M. Fliedner, and A. Scholl, "A classification of assembly line balancing problems," *Eur. J. Oper. Res.*, vol. 183, no. 2, pp. 674–693, Dec. 2007.
[5] O. Battaïa and A. Dolgui, "Reduction approaches for a generalized line balancing problem," *Comput. Oper. Res.*, vol. 39, no. 10, pp. 2337–2345, 2012.
[6] N. T. Thomopoulos, "Mixed Model Line Balancing with Smoothed Station Assignments," *Manage. Sci.*, vol. 16, no. 9, pp. 593–603, 1970.
[7] H. Gökċen and E. Erel, "Binary Integer Formulation for Mixed-Model Assembly Line Balancing Problem," *Comput. Ind. Eng.*, vol. 34, no. 2, pp. 451–461, Apr. 1998.
[8] Z. D. Zhang and H. Sharifi, "Towards theory building in agile manufacturing strategy - A taxonomical approach," *IEEE Trans. Eng. Manag.*, vol. 54, no. 2, pp. 351–370, 2007.
[9] I. Kucukkoc and D. Z. Zhang, "Simultaneous balancing and sequencing of mixed-model parallel two-sided assembly lines," *Int. J. Prod. Res.*, vol. 52, no. 12, pp. 3665–3687, 2014.
[10] Ş. Akpinar and A. Baykasoğlu, "Modeling and solving mixed-model assembly line balancing problem with setups. Part I: A mixed integer linear programming model," *J. Manuf. Syst.*, vol. 33, no. 1, pp. 177–187, Jan. 2014.
[11] B. Yagmahan, "Mixed-model assembly line balancing using a multi-objective ant colony optimization approach," *Expert Syst. Appl.*, vol. 38, no. 10, pp. 12453–12461, Sep. 2011.
[12] L. Tiacci, "Simultaneous balancing and buffer allocation decisions for the design of mixed-model assembly lines with parallel workstations and stochastic task times," *Int. J. Prod. Econ.*, vol. 162, pp. 201–215, Apr. 2015.
[13] I. Kucukkoc and D. Z. Zhang, "Mathematical model and agent based solution approach for the simultaneous balancing and sequencing of mixed-model parallel two-sided assembly lines," *Int. J. Prod. Econ.*, vol. 158, pp. 314–333, 2014.
[14] P. M. Vilarinho and A. S. Simaria, "ANTBAL: An ant colony optimization algorithm for balancing mixed-model assembly lines with parallel workstations," *Int. J. Prod. Res.*, vol. 44, no. 2, pp. 291–303, 2006.
[15] A. Hamzadayi and G. Yildiz, "A simulated annealing algorithm based approach for balancing and sequencing of mixed-model U-lines," *Comput. Ind. Eng.*, vol. 66, no. 4, pp. 1070–1084,

Dec. 2013.

[16] N. T. Thomopoulos, *Assembly Line Planning and Control*. Springer International Publishing, 2013.

[17] N. Boysen, M. Fliedner, and A. Scholl, “The product rate variation problem and its relevance in real world mixed-model assembly lines,” *Eur. J. Oper. Res.*, vol. 197, no. 2, pp. 818–824, Sep. 2009.

[18] Ş. Akpinar and A. Baykasoğlu, “Modeling and solving mixed-model assembly line balancing problem with setups. Part II: A multiple colony hybrid bees algorithm,” *J. Manuf. Syst.*, vol. 33, no. 4, pp. 445–461, Oct. 2014.

[19] P. Chutima and P. Chimklai, “Multi-objective two-sided mixed-model assembly line balancing using particle swarm optimisation with negative knowledge,” *Comput. Ind. Eng.*, vol. 62, no. 1, pp. 39–55, Feb. 2012.

Modelling of Two-sided Assembly Line Balancing Problem with Resource Constraints

Modelling of Two-sided Assembly Line Balancing Problem with Resource Constraints

Muhammad Razif Abdullah Make[1], **Mohd Fadzil Faisae Rashid**[1,*], **Muhamad Magffierah Razali**[1]
Manufacturing Focus Group, Faculty of Mechanical Engineering, Universiti Malaysia Pahang, 26600, Pekan, Pahang, Malaysia.

Email: ffaisae@ump.edu.my

Abstract. Two-sided assembly line balancing (2S-ALB) problems is practically useful in improving the production of large-sized high-volume products. Many published papers have proposed various approaches to balance this well-known ALB problem. However, little attention is given in formulating the 2S-ALB problems. In this paper, 2S-ALB is modelled with four different objective functions comprising minimization of workstations, mated-workstation, idle time and resource constraints. In different with existing model, this paper also considers resource constraint with a mathematical modelling formulation in solving the 2S-ALB problems. The modelling procedures are present for each objective functions with a simple 2S-ALB example problem. Then, the anticipated performance solution is obtained from the test problem.

1. Introduction

The assembly line was firstly introduced in manufacturing system by Henry Ford in 1913. The aim of his idea is to develop the automobile plants for mass production used, with some sort of lines customization. Basically assembly line will consist of a set of a workstation that connected each other in linear fashion. Considering the much numbered of task and workstation presented in assemblies, the problem balancing approach are made. Assembly Line Balancing (ALB) problem formally presented with optimization objective and capable of enhancing the production rate or even the assembly lines. In 1993 Bartholdi has successfully become a pioneer initiating a new type of ALB problem consists with two sides of assemblies, known as Two-sided Assembly Line Balancing (2S-ALB) problem [1].

The 2S-ALB is classified under the General Assembly Line Balancing (GALB) problem, besides the Simple Assembly Line Balancing (SALB) problem. Further research on 2S-ALB is continued in [2-4] studies after Bartholdi in 1993, since the importance of 2S-ALB is highly recognized in the manufacturing industries. Importantly as 2S-ALB for being highlighted through its strength and potency in the manufacturing of a large-sized product. It is definitely managed to improve the assembly productivity throughout the installation. Compared to SALB, with single line operation the 2S-ALB remarkably advantageous since able to shortens the assembly line, save spaces, reduces the cost of tools and fixtures, reduces the throughput time and fit to cut the material handling [1]. These clearly give a good reason for utilizing the 2S-ALB into the assembly line.

In practice, 2S-ALB has firstly proposed in 1993 by Bartholdi in using an interactive computer program with the first fit heuristic method [1]. The successful study focusing on the large-sized product has continued by Kim, Kim et al. (2000) by adopting a genetic algorithm (GA) method [2].

Then, in 2001 a study on maximizing the work relatedness and slackness using a group assignment procedure has been developed by Lee, Kim et al. in solving the 2S-ALB problem [3]. Meanwhile, in 2008 another meta-heuristic approach for 2S-ALB has been proposed by Baykasoglu and Dereli [4]. They have applied an ant colony optimization (ACO) method in the 2S-ALB problem for a domestic product. The highly potential 2S-ALB problem has successfully influenced the other researcher to start a new stage in ALB problem studied. A complex combination with the 2S-ALB problem has been reported through several presented types of research. Such in Simaria and Vilarinho (2009), Özcan and Toklu (2009), Chutima and Chimklai (2012) and Yuan, Zhang et al. (2015) who previously proposed the combination of mixed-model and two-sided ALB problem adopting a different meta-heuristic method as their solving approach [5-8]. Meanwhile, in Kucukkoc and Zhang (2014) a mixed-model parallel two-sided assembly line balancing problem with an agent-based ant colony optimization approach was firstly introduced [9]. The idea of combining 2S-ALB problem with the other ALB model seem totally change the assembly configuration, however it effectively gives more favourable advantages.

This paper is focused on presenting the mathematical modelling for minimizing a multi-objective function implementing the 2S-ALB optimization problem. Our 2S-ALB is a general model emphasize four objective functions, comprising the minimization number of the workstation, mated-workstation, idle time and resource constraint. Besides, the particular feature of 2S-ALB also will be highlighted with a certain model as the numerical examples.

2. 2S-ALB Problem modelling

The 2S-ALB problem has been effectively studied using a specified method since last two decades. The recognition of 2S-ALB problem crucially influences numerous researcher to look forward adopting this ALB problem in their studies [10, 11]. The workstation of the one-sided assembly line is prepared in the linear line of the production system. Figure 1 illustrates the example of the one-sided assembly line with several numbers of the workstation. Each workstation will have certain assigned task and must be completed before moving to another workstation for another task or job. The product will be prepared after all tasks on every workstation are executed.

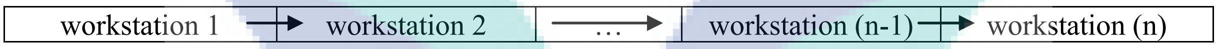


Figure 1. One-sided assembly line

However, a different arrangement of workstation has been offered by two-sided assembly line. The illustration of two-sided assembly line has been depicted in figure 2. For two-sided assembly line the length of workstation logically is shorter than a one-sided assembly line [12, 13], since the tasks and the workstation is divided into two parallel lines. Therefore, it rationally shows some space efficient of the two-sided assembly line. A shorter line is able to reduce the material handling cost, besides the tools and fixture might be placed in the middle between the two parallel workstations (mated-workstation) [1, 12-14]. Furthermore, the two-sided assembly line allowed the task to be executed at the same time for both parallel workstations (left and right) along the assembly lines. However, the precedence relation for each assigned tasks should seriously be assessed before being executed on a specific workstation. The precedence relation among tasks will be discussed in the following subsection.

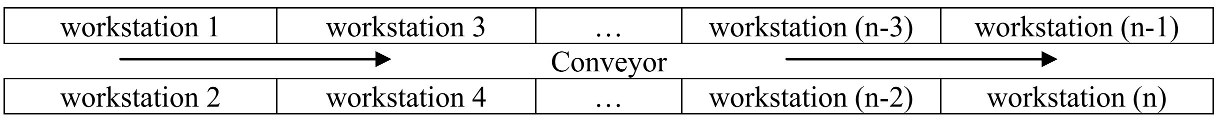


Figure 2. Two-sided assembly line

On the other hand, Bartholdi also expressed the difficulty in permitting the two parallel mated-workstations to work for the same time [1]. The idle workstation will turn into a problem in order to attempt a balanced assembly line. Besides, for each assigned tasks it also possesses with a different particular duration that seems hard to be aligned and balance the assembly lines. Therefore, a comprehensive study is needed in making the two-sided assembly line be balanced and completely able to optimize the ALB problems.

2.1 Precedence relation

A precedence relation graph is constructed to shows an overview of the tasks to be performed. Figure 3 presents the precedence relation graph with some details. As shown below (figure 3) the circle indicates as the assigned task and each task is associated with a processing time and operational direction label (*t, d*). While the arrows linked represent the relation between each task.

In two-sided assembly lines, the operational direction is allowed to be carried out on the same parallel workstation of both sides (left and right) on the same product. Due to the use of both sides of the lines, the additional operating direction has been classified into three different groups and must be obeyed. Three classified group of the operational direction: the left side (L), the right side (R), and either side (E). For left and right side the task execution is absolute and need to be implemented in the following position. Meanwhile, for either side, the execution of tasks is practicable to be performed on any side of the workstation. All the information concerning the assembly processes is disclosed on top of each precedence task

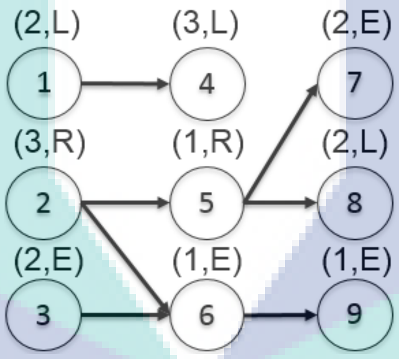


Figure 3. Precedence relation graph (9 task example problem)

2.2 Computational data presentation

From the precedence diagram in figure 3, the relation between each task is important to be transformed into a digital format. The aim is to establish the precedence matrix that could be understood by the computer. Table 1 indicates the information of the precedence graph that being changed into a language for the computational purpose. This precedence matrix consists of ones and zeros values. The value of one indicates the precedence between task i and the next task j. While, zero means no precedence between both task i and j.

Table 1. Precedence matrix

i/j	1	2	3	4	5	6	7	8	9
1	0	0	0	1	0	0	0	0	0
2	0	0	0	0	1	1	0	0	0
3	0	0	0	0	0	1	0	0	0
4	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	1	0
6	0	0	0	0	0	0	0	0	1
7	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0

Besides, the information of time and operational direction (*t, d*) on the precedence relation graph also needs to be modified for the computational purpose. The data matrix on Table 2 presents the example of detail data of assembly information, in which each value brings a particular meaning. The time column on Table 2 represents the processing period of each task, while on side column three different values indicate different sides of the operational direction (1=left side, 2=either side, 3=right side). The other columns denoted as resource constraint data details.

Table 2. Data matrix

task	time	side	resource constraint				
1	2	1	1	2	0	0	0
2	3	3	3	0	0	0	0
3	2	2	2	3	0	0	0
4	3	1	1	0	0	0	0
5	1	3	3	0	0	0	0
6	1	2	2	3	0	0	0
7	2	2	1	2	3	0	0
8	2	1	2	0	0	0	0
9	1	2	1	3	0	0	0

2.3 Multi-objective modelling and execution
In this paper, the 2S-ALB is evaluated with four objectives; to minimize the number of the workstation, mated-workstation, idle time and resource constraint. The general assumptions of the problem are as follows: -

- The workstation numbers are not fixed and no absence of any workstation either left or right along the assembly line.
- The maximum operational cycle time is fixed and could not be exceeded.
- Three different operational directions of the left side, either side and the right side are presented in numerical order.
- The assembly operation /task can be started at the same time on both sides for every mated-workstation.
- The low level of operator skills is ignored, for the assembly tasks will be assembled at the same rate and pace.
- Each workstation can only be assigned by a single operator.

Considering the above assumptions, the parameters that are involved in permitting all the objective functions are as follow.

J	: number of mated-workstation $j = 1, 2, \ldots, J$
I	: number of one-sided workstation $i = 1,2, \ldots, I$
F	: 1, if there is any space availability on the operating time, otherwise, 0
N	: number of resource utilization $n = 1,2, \ldots, N$
X_{ms}	: 1, if mated-workstation j is utilized for both side of the line, otherwise, 0
Y_s	: 1, if mated-workstation j is utilized for only one side of the line, otherwise, 0
m_t	: maximum processing time $t = 1,2, \ldots, T$
r_t	: operational time of the task on the workstation j
p_v	: maximum gap value in space availability
q_v	: minimum gap value in space availability
R_s	: 1, if resource is utilized in workstation j, otherwise, 0

The mathematical model of the problem can be formulated as follows.

$$f1 = min \sum_{j=1}^{J} X_{ms} \tag{1}$$

$$f2 = min \sum_{j=1}^{J} 2JX_{ms} + \sum_{i=1}^{I} Y_s \tag{2}$$

$$f3 = min \sum_{t=1}^{T} (m_t - r_t) + \sum_{t=1}^{T} F(p_v - q_v) \tag{3}$$

$$f4 = min \sum_{n=1}^{N} R_s \tag{4}$$

The objective function in equation (1) aims to minimize the number of mated-workstation. Where X_{ms} shows the sum of utilized number of mated-workstations. Objective function (2) indicates the other aims of minimizing the number of utilized workstations. Meanwhile, objective function (3) represent the minimization of idle time value, considering the operational processing time m_t,r_t and the space time availability p_v, q_v on each workstation. Then, in objective function (4) the minimization of resource constrain is described with the summation of resource utilization R_s.

3. Numerical example

This section presents a numerical example to explain the objective functions in section 2. For this purpose, the example in figure 3 is considered. In this example, the cycle time is fixed with 6 units of times. Therefore, each workstation completely allowed to have the unlimited assign task but it should not exceed 6 unit of cycle time. Figure 4 illustrates the task distribution layout of 2S-ALB problems. In distributing the tasks to both sides of the lines formerly the assembly sequence associated with tasks will firstly be generated. As the example, one assign sequence is proposed comprising of 9 assembly tasks: 2,1,3,6,5,4,9,8,7. Then, according to each time and side data (*t, d*), the task will be assigned to the appropriate workstation. However, in filling the job for workstation sometimes the precedence will turn as a constraint, thus causing an idle time. The shaded area represents as idle time with no assigned task /job. The task distribution and layout will be different if the generated sequence is changed. Means that, the total idle time will also change, either turned into a longer or shorter period of time.

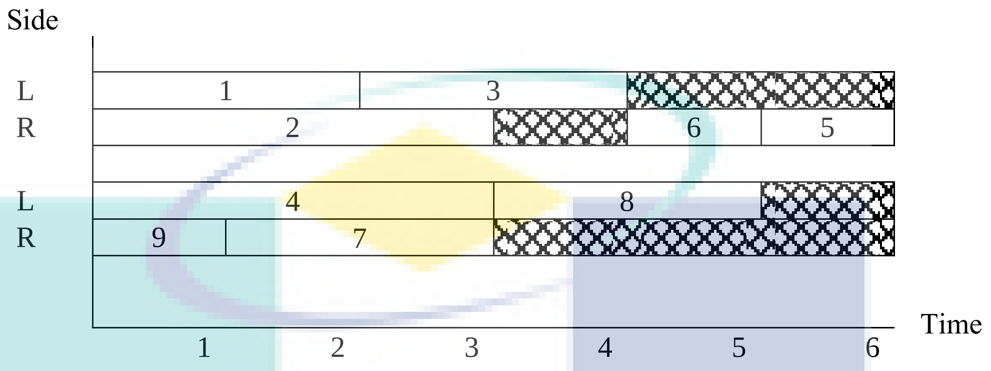


Figure 4. 2S-ALB task distribution

The analysis and evaluation of this 2S-ALB problem modelling also depicted from figure 4. Along the assembly line, four different aim will be calculated associate to the 9 assigned task (problem example) from figure 3.

3.1 Minimize the number of mated-workstation

The particular equation of this aims is subjected to equation (1), while the assembly line illustration is depicted in figure 4. The calculation of the first objective function $(f1)$ of minimizing the number of mated-workstation is presented below. The two-sided workstation consists of left (L) and right (R) is recognized as mated-workstation. Thus the computation of $f1$ is calculated by the summation of utilized mated-workstation (j). It will be summed before multiplied by X_{ms} condition, either 1 or 0. As the sum of mated-workstation $(1 + 1)$, then it is multiplied by 1 as both side utilization of the workstation.

$$f1 = min \sum_{j=1}^{J} X_{ms} \tag{1}$$

$$f1 = (1 + 1) \text{ x } 1$$
$$f1 = 2$$

2.4 Minimize the number of workstation

While the second objective function is subjected to equation (2). The assembly line illustration also depicted from figure 4. The totalled of mated-workstation will be multiplied by 2 and X_{ms} condition. Then it should be added to the total of one-sided utilized workstation as its evaluation. As J is equal to $f1$, the number of mated-workstation is set to be two. The value of J will be multiplied by 2 and 1 as the condition of both sided workstation utilization, $(2(2) \text{ x } 1)$. After that, the total of one /single sided utilized workstation will be summed with the previous value for $f2$ values. Y_s is the condition for single sided utilized workstation, either 1 or 0.

$$f2 = min \sum_{j=1}^{J} 2JX_{ms} + \sum_{i=1}^{I} Y_s \tag{2}$$

$$f2 = [2(2) \text{ x } 1] + [0(0)]$$
$$f2 = 4$$

2.5 Minimize idle time

Idle time is a wasted duration with no assigned task. Associated as a void space its graphically shown in figure 4 with shaded area. The best balanced line approximately has equal processing time. Therefore, minimizing the idle time is able to enhance the assembly operation line. The idle time formulation is presented in equation (3). The total subtraction of maximum processing time (cycle time) and the task time of each workstation, will be added together with the space availability on every workstation if any. The value of maximum processing time will be fixed on 6 unit of time before subtracting with the operation task time for each workstation (4,6,5,3). As the example on the first workstation with 4 unit of processing time, $(6 - 4)$. Then, the total needs to add with another total gap /space along the assembly line for each workstation if any, such on the second workstation $(4 - 3)$. The deduction of maximum and minimum gap space will be summed for every workstation and multiplied with the F condition. The value will be totaled and come out as $f3$ value

$$f3 = min \sum_{t=1}^{T} (m_t - r_t) + \sum_{t=1}^{T} F(p_v - q_v) \tag{3}$$

$$f3 = [(6-4) + (6-6) + (6-5) + (6-3)] + [(4-3) \text{ x } 1]$$
$$f3 = 7$$

2.6 Minimize resource constraint

As shown in Table 3, the tasks and resource have divided into four number of workstation. Since that, the resource utilization of machine and tools seem is repeated even for the same workstation. The minimization of the resource used significantly help in reducing the cost of tools and fixture. For this example, it has 3 types of fixtures that used in completing all of the tasks. The resource row indicated the fixtures used with one type of machine (M1) and two types of tools (T1 and T2). However, the same machine and tools which located in the same workstation practically turned into a waste. The calculation on minimizing the resource constraint is presented below. The total resource used for every workstation is calculated emphasizing machine and tools. Such in workstation 1 possessing with 1 type of machine and 2 type of tools. Therefore, as the total 3 resource is considered from workstation 1. The resource used for another workstation also calculated to be summed before its multiplied with R_s condition which either 1 if the resource is utilized in the workstation j or 0, otherwise.

Table 3. Resource utilization

Workstation (w)	1		2			3		4	
Tasks (s)	1	3	2	6	5	4	8	9	7
Resource (Machine & tool used), (r)	M1 T1	T1 T2	T2	T1 T2	T2	M1	T1	M1 T2	M1 T1 T2
Machine (m)	M1		-			M1		M1	
Tool (t)	T1,T2		T1, T2			T1		T1,T2	

$$f4 = min \sum_{n=1}^{N} R_s \tag{4}$$

$$f4 = (3 + 2 + 2 + 3) \text{ x } 1$$
$$f4 = 10$$

4. Discussion

A slight different have shown in 2S-ALB compared to one-sided assembly. The deterministic performance time with a specific order of precedence tasks leads to the idle time formation. Previously in one-sided assembly, the ordered task and workstation are presented on a single straight line (figure 1). The preferred task will be assign to the workstation one to another until the product is complete and ready to use. But compared to two-sided assembly a short duration of idle time might be available on a certain workstation. This happens because of the precedence preferred assigned side along the assembly line. Although the idle time is issued in the 2S-ALB problem, it is believed more effective for improving the productivity. Such as able to avoid the excessive workload, minimize wasted or over processing leads to excess inventory, and have less processing time with higher production rate compared to one-sided assemblies.

Another different initiated from this 2S-ALB modelling paper is the resource constraint utilization. Table 3 demonstrates the resource utilization on every workstation with assigned task. The repeated used of the resource (machine and tools) in executing the task along the assembly line could be reduced by implementing the 2S-ALB operation. By installing 2S-ALB configuration the resource utilization of machine and tools used definitely will be minimized when that fixture are placed in the middle (left and right) between two mated-workstation. The minimum number of machine tools and fixture surely reduced the maintenance cost apart of adding the same machine and tools. Over numerous benefit and advantages, the 2S-ALB is known from the literature to successfully provide the high efficiency in operational lines. Practically the accomplishment of 2S-ALB has abundantly proven in the assembly of a large-sized product, such bus and trucks [2] and automobile [3]. Even so, the success has been continued for a domestic product and is presented in Baykasoglu and Dereli in 2008 studies [4]. From all of the success, it shown the 2S-ALB is potentially could be used in another field of studies, such printing and food processing industry. This could potentially enhance the 2S-ALB implementation in other various fields.

5. Conclusion

In this paper, the model of two-sided assembly line balancing (2S-ALB) problem was discussed. This study undertaken to design a multi-objective problem with four different objective functions. The minimization number of workstation, mated-workstation, idle time and resource constraint was introduced based on the real world problem. Further on the 2S-ALB modelling evaluation of four aims is presented in details with a particular equation and calculation. The obtained results have described the 2S-ALB features. In addition, the comparison of the performance of 2S-ALB modelling is also stated in this study to give some clear explanation with its advantages. From the results of this modelling, it shows the 2S-ALB could have a better performance in optimization.

For future work, we might extend the proposed model of the 2S-ALB problem for another optimization phase. Its involve with the implementation of computational optimization approach such GA and ACO method. This modelling also fits for other optimization problem in the area dealing with other objectives. Besides, a larger 2S-ALB problem, with more number of assign task associate to the assemblies is necessary to discuss in the future.

Acknowledgments
The authors would like to acknowledge the Ministry of Higher Education, Malaysia and Universiti Malaysia Pahang for supporting this research under the FRGS grant RDU140103.

References

[1] Bartholdi, J.J., Balancing two-sided assembly lines: a case study. *International Journal of Production Research,* 1993. **31**(10): p. 2447-2461.
[2] Kim, Y.K., Y. Kim, and Y.J. Kim, Two-sided assembly line balancing: A genetic algorithm approach. *Production Planning & Control,* 2000. **11**(1): p. 44-53.
[3] Lee, T.O., Y. Kim, and Y.K. Kim, Two-sided assembly line balancing to maximize work relatedness and slackness. *Computers & Industrial Engineering,* 2001. **40**(3): p. 273-292.
[4] Baykasoglu, A. and T. Dereli, Two-sided assembly line balancing using an ant-colony-based heuristic. *The International Journal of Advanced Manufacturing Technology,* 2008. **36**(5-6): p. 582-588.
[5] Simaria, A.S. and P.M. Vilarinho, 2-ANTBAL: An ant colony optimisation algorithm for balancing two-sided assembly lines. *Computers & Industrial Engineering,* 2009. **56**(2): p. 489-506.
[6] Özcan, U. and B. Toklu, Balancing of mixed-model two-sided assembly lines. *Computers & Industrial Engineering,* 2009. **57**(1): p. 217-227.
[7] Chutima, P. and P. Chimklai, Multi-objective two-sided mixed-model assembly line balancing using particle swarm optimisation with negative knowledge. *Computers & Industrial Engineering,* 2012. **62**(1): p. 39-55.
[8] Yuan, B., et al., An effective hybrid honey bee mating optimization algorithm for balancing mixed-model two-sided assembly lines. *Computers & Operations Research,* 2015. **53**: p. 32-41.
[9] Kucukkoc, I. and D.Z. Zhang, Simultaneous balancing and sequencing of mixed-model parallel two-sided assembly lines. *International Journal of Production Research,* 2014. **52**(12): p. 3665-3687.
[10] Abdullah Make, M.R., M.F.F. Ab. Rashid, and M.M. Razali, A review of two-sided assembly line balancing problem. *The International Journal of Advanced Manufacturing Technology,* 2016: p. 1-21.
[11] Rashid, M.F.F., W. Hutabarat, and A. Tiwari, A review on assembly sequence planning and assembly line balancing optimisation using soft computing approaches. *The International Journal of Advanced Manufacturing Technology,* 2011. **59**(1): p. 335-349.
[12] Khorasanian, D., S.R. Hejazi, and G. Moslehi, Two-sided assembly line balancing considering the relationships between tasks. *Computers & Industrial Engineering,* 2013. **66**(4): p. 1096-1105.
[13] Tuncel, G. and D. Aydin, Two-sided assembly line balancing using teaching–learning based optimization algorithm. *Computers & Industrial Engineering,* 2014. **74**: p. 291-299.
[14] Purnomo, H.D. and H.-M. Wee, Maximizing production rate and workload balancing in a two-sided assembly line using Harmony Search. *Computers & Industrial Engineering,* 2014. **76**: p. 222-230.

CHAPTER 4

RESULTS AND DISCUSSIONS

Computational experiment were conducted separately for three version of ALB problems. Some of the result was published, while some of the papers were still under review as follow:

1. Jusop, M. and Ab. Rashid, M.F.F. (2016),"Optimisation of Assembly Line Balancing Type-E with Resource Constraints using NSGA-II", Key Engineering Materials, vol. 701, pp 195-199.
2. Jusop, M. and Ab. Rashid, M.F.F. (2017), "Optimization of Assembly Line Balancing with Resource Constraint using NSGA-II: A Case Study", International Journal of Applied Engineering Research 12 (7), 1421-1426.
3. Kamaruddin, N.H. and Ab. Rashid, M.F.F. (2017), "Assembly Line Balancing with Resource Constraints using New Rank-Based Crossovers", ICADME 2017 Conference
4. Razali, M.M., Ab. Rashid, M.F.F. and Abdullah Make, M.R. (2017), "Optimization of Mixed-Model Assembly Line Balancing Problem with Resource Constraints", Journal of Mechanical Engineering (Under review).
5. Abdullah Make, M.R., Ab. Rashid, M.F.F. and Razali, M.M. (2017), "Optimization of Two-Sided Assembly Line Balancing with Resource Constraints using Modified PSO", International Journal of Advanced manufacturing Technology (Under review).

Key Engineering Materials
ISSN: 1662-9795, Vol. 701, pp 195-199
doi:10.4028/www.scientific.net/KEM.701.195
© 2016 Trans Tech Publications, Switzerland

Submitted: 2015-10-12
Revised: 2016-03-17
Accepted: 2016-03-29
Online: 2016-07-10

Optimisation of Assembly Line Balancing Type-E with Resource Constraints using NSGA-II

MASITAH JUSOP[1,a]* , MOHD FADZIL FAISAE AB RASHID[1,b]

[1]Faculty of Mechanical Engineering, Universiti Malaysia Pahang, 26600 Pekan, Pahang, Malaysia

[a]masitahjusop@yahoo.com, [b]ffaisae@ump.edu.my

Keywords: Assembly Line Balancing, Type-E, Resource constraints, NSGA-II.

Abstract. Assembly line balancing of Type-E problem (ALB-E) is an attempt to assign the tasks to the various workstations along the line so that the precedence relations are satisfied and some performance measures are optimised. A majority of the recent studies in ALB-E assume that any assembly task can be assigned to any workstation. This assumption lead to higher usage of resource required in assembly line. This research studies assembly line balancing of Type-E problem with resource constraint (ALBE-RC) for a single-model. In this work, three objective functions are considered, i.e. minimise number of workstation, cycle time and number of resources. In this paper, an Elitist Non-dominated Sorting Genetic Algorithm (NSGA-II) has been proposed to optimise the problem. Six benchmark problems have been used to test the optimisation algorithm and the results are compared to multi-objective genetic algorithm (MOGA) and hybrid genetic algorithm (HGA). From the computational test, it was found NSGA-II has the ability to explore search space, has better accuracy of solution and also has a uniformly spaced solution. In future, a research to improve the solution accuracy is proposed to enhance the performance of the algorithm.

Introduction

Assembly Line Balancing (ALB) refers to the decision problem of optimally partitioning the assembly task among the workstations with respect to some objectives [1]. This research studies ALB-E with resource constraints, such as machine, tool and worker that are needed to conduct the assembly process for a particular product. In a real-world problem, there are limited numbers of machines and tools that can be used to assemble a certain product. In order to maximise the resource utilisation in assembly line, the number of resources must be taken into consideration while assigning the task in any workstation. Traditional Genetic Algorithm may give an accurate solution but it may not be able to search for an optimal solution for a more complex problem. For the purpose of finding one single optimal solution for multi-objective optimisation problem, Deb et al [2] suggested to convert the problem into a single-objective optimisation problem. In this paper, an Elitist Non-dominated Sorting Genetic Algorithm (NSGA-II) has been proposed to optimise the problem. ALB-E is a type of NP-hard optimisation problems with an extremely large number of feasible solutions [3-6]. Advanced approach of algorithm is necessary to solve large-scale problems. The NSGA-II was developed to accommodate multi-objective optimisation problems. Other than the capability to solve a more complex and real world multi-objective optimisation problem, the other reason for choosing NSGA-II is its ability to find a better spread of solutions [7].

In summary, it can be concluded that only a few researcher consider ALB-E in their research due to the complexity of the problem. To the best of author knowledge, no attempt has been carried out to implement NSGA-II in ALB-E with resource constraint (ALBE-RC). However, it is important to be concerned about these constraints because of limited number of resources in industry.

ALBE-RC Modelling

This section presents an approach for the optimisation of ALBE-RC problem through a simple representation diagram. The problem representation step starts with establish a liaison matrix which is used to generate feasible assembly sequences. Once the liaison matrix has been established, DeFazio's question and answer procedure is applied for the purpose of identifying the existence of precedence relations in assembly tasks. There are two questions that must be considered while evaluating each of the assembly task: (i) what tasks must be done prior to doing task *i*? (ii) what tasks must be left to be done after doing task *i*? [8]. The precedence graph mapping later can be formed after DeFazio's question and answer is applied. Then, the assembly data for proposed representation can be tabulated in n×4 table where n is the number of assembly task. The first column t denotes task time whereas the other three columns R_A, R_B and R_C correspondingly represent type of resource used. Finally, a feasible assembly sequence need to be evaluated according to the objective functions. In this study, there are 3 objective functions that have to be measured: (i) minimise number of workstation (ii) minimise the cycle time (iii) minimise number of resources.

NSGA-II for ALBE-RC

Deb et al [2] first introduced an Elitist Non-dominated Sorting Genetic Algorithm (NSGA-II) as an improved version of NSGA. NSGA-II procedure starts with initializing a random population P_i of size N. The parent population P_i is sorted by the non-dominated sorting approach. New population Q_i of size N is obtained by selection, crossover and mutation. New population R_i of size 2N is formed by combining population P_i and Q_i ($R_i = P_i \cup Q_i$). The R_i population is sorted by the non-dominated sorting approach. Population belonging to the best non-dominated set F_i is filled as a new population P_{i+1}. The counter i is set, i=1. The remaining non-dominated solutions are sorted again. This process continues until all the solutions are filled according to the non-domination level. The crowding distance of each solution with different non-domination levels is calculated. The population is sorted in descending order of magnitude of the crowding distance values. The flowchart of NSGA-II is shown in Fig. 1.

Computational Experiment. Six benchmark problems are used to test the optimisation algorithm. All the test problems except the problem by Ağpak & Gökçen [9] have been modified by including the resources but the original informations are preserved. Small size problems are taken from Ağpak & Gökçen [9] and Ponnambalam et al [10] whereas medium and large problems are available on an online database for assembly line balancing research: www.assembly-line-balancing.de. The proposed algorithm is coded in Matlab 7.14 and the experiments are executed on a Windows 8, Intel® Core™ *i*5-4210U CPU 1.70 GHz with 4 GB of RAM. The following parameters have been used to run the experiments: Population size, 20; Number of generations, 200; Crossover probability, 0.8; Mutation probability, 0.3.

Comparison algorithms. Genetic algorithms (GAs) are mainly used by researchers for optimisation of large and complex problems specifically in ALB problem [11-13]. In order to measure the performance of the optimisation algorithm, three genetic algorithms, namely Non-dominated sorting genetic algorithm (NSGA-II), Multi-objective genetic algorithm (MOGA) and Hybrid genetic algorithm (HGA) are tested with six test problems taken from open literature. Multi-objective Genetic Algorithm (MOGA) has been introduced by Fonseca and Fleming [14] in 1993. MOGA has the abilities to find a diverse set of non-dominated solutions and to explore a nearly-true to the optimal set of solutions [15, 16]. It has also been widely used in real-world optimisation problems to solve assembly line balancing problems. In the paper presented by Ponnambalam et al [10], the researchers concluded that MOGA take more time in finding the global optimal solutions. A hybrid GA has been proposed by Chen et al [17] to solve the assembly line planning problem. The proposed GA is able to search for many feasible solutions in a short time. Genetic algorithm is an approach used in finding an optimal solution for a complex optimisation problem [12]. Valls et al [18] stated that hybrid genetic algorithm is high in quality and is a fast algorithm that is better than all state-of-the-art algorithms.

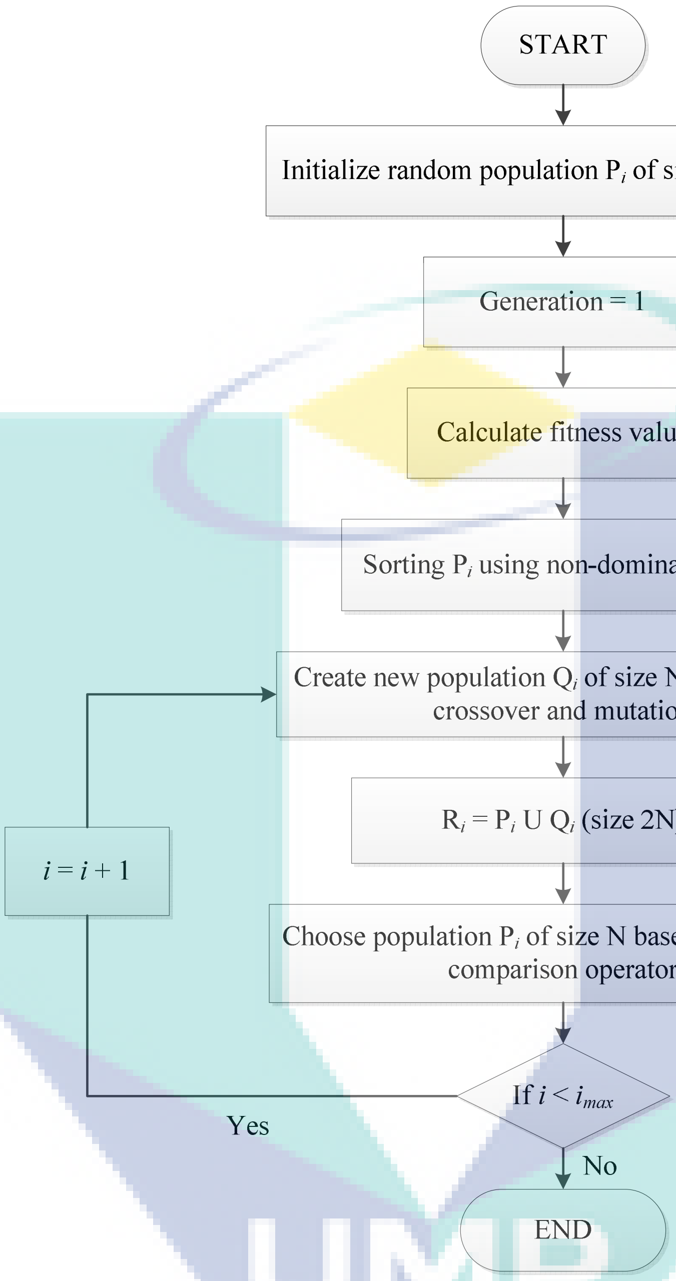

START
Initialize random population P_i of size N (P_i, $i = 0$)
Generation = 1
Calculate fitness value
Sorting P_i using non-dominated sort
Create new population Q_i of size N by selection, crossover and mutation
$R_i = P_i \cup Q_i$ (size 2N)
Choose population P_i of size N based on crowded-comparison operator
If $i < i_{max}$
Yes
No
$i = i + 1$
END

Figure 1 Flowchart of NSGA-II

Results and Discussion

To evaluate the performance of the proposed algorithms, five performance indicators are measured. The results of the performance measure of the algorithms are presented in Table 1 and the discussions are given in the following sections.

Performance Indicators. The number of non-dominated solution (*NDS*) is measured to identify the ability of the algorithm to explore the search space. Meanwhile, Error Ratio (*ER*) and Generational Distance (*GD*) metrics measure the accuracy of solution. *Spacing* metric measures the uniformity of solution whereas maximum spread ($Spread_{max}$) is evaluated in order to determine the spread of solution. Details on these performance metrics can be referred in [15]. The results from the computational tests show that NSGA-II performs better in finding the non-dominated solutions for all six problems. It can be concluded that NSGA-II has the ability to explore the search space compared to MOGA and HGA. Small *ER* and small *GD* will increase the accuracy of the solution. In comparing with MOGA and HGA, NSGA-II has the smallest *ER*. Therefore, it can be stressed that NSGA-II has better accuracy of solution. In addition to the result of Error Ratio, the result of *GD* also showed that the performance of NSGA-II dominates the performance of MOGA and HGA in having high accuracy of solution. From Table 1, it is apparent that NSGA-II has the smallest value of *GD* for all problems compared with MOGA and HGA

Table 1 Results of performance mesure of the algorithms MOGA, HGA and NSGA-II

Problem	Algorithm	*NDS*	*ER*	*GD*	*Spacing*	$Spread_{max}$
Problem 1	MOGA	1	0.5	0.5	0	1.7321
	HGA	1	0.5	0.5	0	1.7321
	NSGA-II	3	0	0	0	2.8284
Problem 2	MOGA	3	0	0	0.4714	4.1231
	HGA	3	0	0	0.4714	4.1231
	NSGA-II	3	0	0	0.4714	4.1231
Buxey	MOGA	5	0.1667	0.1667	1.2134	10.247
	HGA	5	0.5	0.6243	0.5	13.9642
	NSGA-II	7	0	0	0.4949	10.247
Kilbridge	MOGA	0	1	1.9255	0.3499	15.3623
	HGA	1	0.8333	0.9714	1.2134	14.0712
	NSGA-II	6	0	0	0.7454	11.5758
Wee-Mag	MOGA	1	0.875	2.903	1.9365	20.3224
	HGA	7	0.5882	1.1236	2.5219	31.7805
	NSGA-II	10	0	0	7.8358	29.0172
Lutz2	MOGA	5	0.6154	0.855	1.8138	33.8526
	HGA	6	0.5714	0.8257	1.2778	34.7131
	NSGA-II	8	0.2	0.2	1.005	27.2029

Out of five performance measures that have been used to compare the algorithms, NSGA-II consistently performed well in three indicators; (i) Number of Non-Dominated Solution, *NDS* (ii) Error Ratio, *ER* (iii) Generational Distance, *GD*. In consequence, it is adequate to prove that the proposed NSGA-II has overcome the performance of the other two comparison algorithms, MOGA and HGA for multi-objective optimisation problem.

Conclusion

This paper focuses on assembly line balancing of Type-E problem with resource constraint (ALBE-RC) for a single-model. In this paper, an Elitist Non-dominated Sorting Genetic Algorithm (NSGA-II) is proposed to optimise the problem. Based on the computational test that has been carried out, NSGA-II has the ability to explore the search space and has better accuracy of solution compared to other algorithms. In future, a research to improve; (i) uniformity of solution (ii) spread of solution are proposed to enhance the performance of the algorithm for all sizes of problem.

Acknowledgement

We acknowledge the Ministry of Higher Education for funding this study through a research grant under RDU140103. The authors would also like to thank Universiti Malaysia Pahang for providing the necessary facilities.

References

[1] M. F. F. Rashid, W. Hutabarat, and A. Tiwari, "A review on assembly sequence planning and assembly line balancing optimisation using soft computing approaches," *The International Journal of Advanced Manufacturing Technology,* vol. 59, pp. 335-349, 2012.

[2] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *Evolutionary Computation, IEEE Transactions on,* vol. 6, pp. 182-197, 2002.

[3] O. Emeke Great and A. Offiong, "Productivity Improvement In Breweries Through Line Balancing using Heuristic Method," *International Journal of Engineering Science & Technology,* vol. 5, 2013.

[4] N. Kriengkorakot and N. Pianthong, "The Assembly Line Balancing Problem," *KKU Enginieering Journal,* vol. 34, pp. 133-140, 2007.

[5] A. C. Nearchou, "Multi-objective balancing of assembly lines by population heuristics," *International Journal of Production Research,* vol. 46, pp. 2275-2297, 2008.

[6] Y. Tsujimura, M. Gen, and E. Kubota, "Solving fuzzy assembly-line balancing problem with genetic algorithms," *Computers & Industrial Engineering,* vol. 29, pp. 543-547, 1995.

[7] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," *Lecture notes in computer science,* vol. 1917, pp. 849-858, 2000.

[8] T. L. D. F. a. D. E. Whitney, "Simplified Generating of All Mechanical Assembly Sequences," *IEEE Journal of Robotics and Automation,* vol. RA-3, No. 6,, 1987.

[9] K. Ağpak and H. Gökçen, "Assembly line balancing: Two resource constrained cases," *International Journal of Production Economics,* vol. 96, pp. 129-140, 2005.

[10] S. Ponnambalam, P. Aravindan, and G. M. Naidu, "A multi-objective genetic algorithm for solving assembly line balancing problem," *The International Journal of Advanced Manufacturing Technology,* vol. 16, pp. 341-352, 2000.

[11] 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, pp. 49-69, 2008.

[12] N. Mohd Razali and J. Geraghty, "Biologically inspired genetic algorithm to minimize idle time of the assembly line balancing," in *Nature and Biologically Inspired Computing (NaBIC), 2011 Third World Congress on*, 2011, pp. 105-110.

[13] R. Ranjan and P. Pawar, "Assembly Line Balancing Using Real Coded Genetic Algorithm," *International Journal of Scientific Research in Computer Science and Engineering,* vol. 2, pp. 1-5, 2014.

[14] C. M. Fonseca and P. J. Fleming, "Genetic Algorithms for Multiobjective Optimization: FormulationDiscussion and Generalization," in *ICGA*, 1993, pp. 416-423.

[15] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*, 1st ed.: John Wiley & Sons, Ltd, 2001.

[16] W. Zhang, M. Gen, and L. Lin, "A multiobjective genetic algorithm for assembly line balancing problem with worker allocation," in *Systems, Man and Cybernetics, 2008. SMC 2008. IEEE International Conference on*, 2008, pp. 3026-3033.

[17] R.-S. Chen, K.-Y. Lu, and S.-C. Yu, "A hybrid genetic algorithm approach on multi-objective of assembly planning problem," *Engineering Applications of Artificial Intelligence,* vol. 15, pp. 447-457, 2002.

[18] V. Valls, F. Ballestin, and S. Quintanilla, "A hybrid genetic algorithm for the resource-constrained project scheduling problem," *European Journal of Operational Research,* vol. 185, pp. 495-508, 2008.

International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 7 (2017) pp. 1421-1426
© Research India Publications. http://www.ripublication.com

Optimization of Assembly Line Balancing with Resource Constraint using NSGA-II: A Case Study

Masitah Jusop
Faculty of Mechanical Engineering
Universiti Malaysia Pahang, Malaysia.

Mohd Fadzil Faisae Ab Rashid
Faculty of Mechanical Engineering
Universiti Malaysia Pahang, Malaysia.

Abstract
Abstract Assembly line balancing type-e problem with resource constraint (ALBE-RC) is an attempt to assign the tasks to a minimal number of workstation with minimum cycle time by considering the resource constraint. Due to rapid growth in manufacturing and limited number of resources in industry, all the tasks that used the same resources will be performed in the same workstation such that the precedence relations are not violated. In this work, an implementation of an elitist non-dominated sorting genetic algorithm (NSGA-II) is proposed to optimise ALBE-RC case study. An industrial case study was conducted in an electronic company and a product known as HM72A-10 series model has been selected for the case study. The results from the optimization shows that all the optimisation parameters i.e. number of workstations, cycle time and number of resources used could be minimised. The improvement of line efficiency also indicated that the optimization results are better that the existing one. The validation from industrial expert provides evidence that the proposed method is applicable and can be implemented for line balancing.

Keywords: Assembly Line Balancing, Type-E, Resource constraint, NSGA-II.

INTRODUCTION

An assembly line is one of manufacturing process comprises of a sequence of workstation in which a set of necessary task to assemble a product are performed. The aim of line balancing is to assign the tasks to an ordered sequence of workstations, such that the precedence relations are not violated and some performance measures are optimised (eg: maximise the line efficiency, minimise the number of workstations and minimise the cycle time). ALB is the decision problem of optimally partitioning the assembly tasks among the workstations related to some objectives [1]. Previous researchers make an assumption that any of assembly task can be performed and can be assigned to any workstation [2-5]. However, in reality each workstation has their own capabilities and specialization.

To the best of author knowledge, there is only a small number of research which consider resource constraint in ALB works [6-9]. Interestingly, none of them consider resource constraint in assembly line balancing type-e (ALB-E) problem itself. Most of previous researcher used traditional GAs as an optimization technique in ALB problem [10-12]. Yet, the implementation of NSGA-II in ALBE-RC has not been given great attention by the researchers [13]. In this work, assembly tasks that used the same resources i.e. machine, tool, and worker will be assigned in one workstation according to the precedence and cycle time constraint. Deb et al. introduced NSGA-II to accommodate a complex and real-world optimization problem for multi-objective function [14, 15]. Besides than incorporate elitism-preserving technique, NSGA-II also has the capabilities to find better solutions.

This paper presents an optimization of assembly line balancing type-E problem with resource constraint (ALBE-RC) on a selected industrial case study by using NSGA-II. The case study was conducted in an electronic company, which produced electronic components in Malaysia.

ELITIST NON-DOMINATED SORTING GENETIC ALGORITHM (NSGA-II)

Elitist Non-dominated Sorting Genetic Algorithm (NSGA-II) is an optimization algorithm developed by Deb et.al in the year of 2000. This algorithm was developed based on evolutionary algorithm, with modification in determining the leader in evolution process. Instead of having the best solution leader, the NSGA-II calculate the Crowding Distance to determine the leader [14-16].

NSGA-II procedure starts with initializing a random population P_i of size N_{pop}. The algorithm is then decoded into feasible sequences using topological sort. The fitness of feasible chromosomes is calculated by evaluate the objective functions. Later, a non-dominated sorting approach is applied to generate Pareto-optimal set. The entire population is sorted using non-dominated sorting approach to identify the non-dominated set $F = (F_1, F_2, \ldots, Fi)$. The parent population is filled with set F according to non-domination rank. If $F > N_{pop}$, the last front will be selected based on higher crowding distance (*CD*). Since NSGA-II used the selection strategy based on crowding distance, it will gives an estimation of the density of selected solutions.

The tournament competition between two random-pair of solutions from parent population is performed to determine the domination rank. The population will be sorted in decreasing rank of level according to each objective function. Solution with better rank is filled in parent pool. Meanwhile, the solution with the same rank but remains in a less crowded area will be selected. The tournament selection is repeated until the parent pool is fully occupied to generate children. New offspring population Q_i of size N_{pop} is generated from P_i by crossover and mutation operators. Later, P_i and Q_i are combined to form new population R_i of size $2N_{pop}$. The NSGA-II procedure is repeated until the termination criteria is met.

As mentioned previously, the aforementioned algorithm implements an elitism-preserving technique. It will ensures that the best solution found in each generation will never be lost until the better solution is discovered [17-19]. The flowchart of NSGA-II is illustrated in Fig. 1.

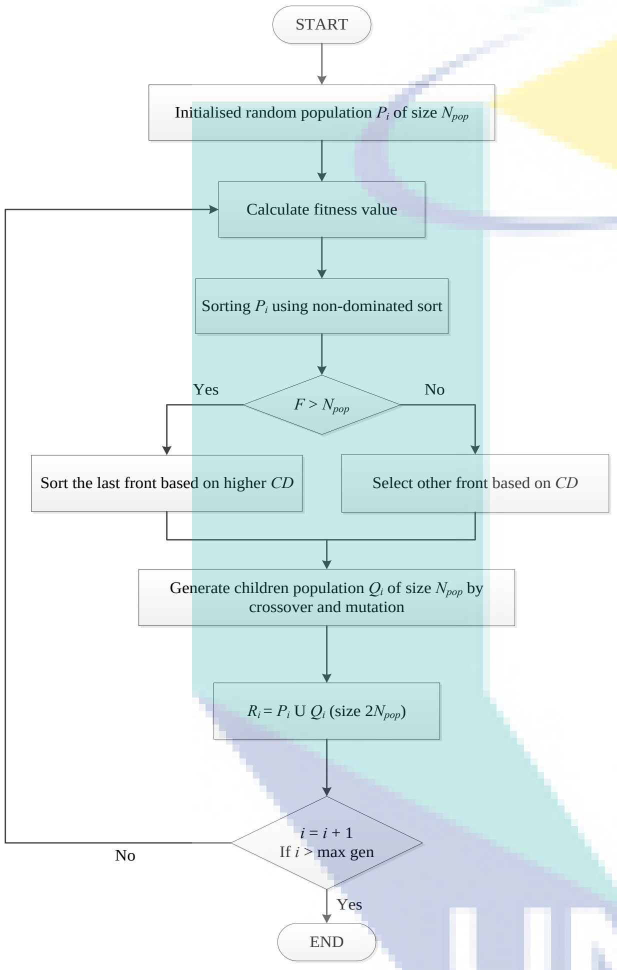

START
Initialised random population P_i of size N_{pop}
Calculate fitness value
Sorting P_i using non-dominated sort
Yes
$F > N_{pop}$
No
Sort the last front based on higher CD
Select other front based on CD
Generate children population Q_i of size N_{pop} by crossover and mutation
$R_i = P_i$ U Q_i (size $2N_{pop}$)
$i = i + 1$
If $i >$ max gen
No
Yes
END

Figure 1. Flowchart of NSGA-II

INDUSTRIAL CASE STUDY

A. Product and Company Background

TT Electronics is a United Kingdom based manufacturer that produces sensing and control for industrial and car makers, advanced components and integrated manufacturing services (IMS). The advanced components provide engineered components solutions such as resistors, power and hybrid devices, magnetics and connectors. The magnetics components are handled by BI Technologies Corporation Sdn. Bhd. that was located in Kuantan, Malaysia. BI Technologies Corporation is wholly owned subsidiary by TT Electronics. Their product design team are focused on custom and semi-custom product based on customers' needs. The products that have been produce by the company are magnetic components, power and signal, inductors SMD (Surface Mount Device) and through hole, molded inductor, and lamination transformer.

For this case study the moulded inductor production section is selected as the product running on the line is a type of single model. Only HM72A-10 series model was running on the line during the data collection. HM72A-10 series is a type of moulded inductor. This class of product is a high power low cost moulded SMD inductor which is typically used in electronic device such as computer. Table I presents the summary of the production line of HM72A-10 series model. A total of 13 workers were assigned in all workstations to perform all the tasks with a number of 30 machines and tools that had been used throughout the process.

Table I: SUMMARY ON HM72A-10 PRODUCTION

Work element		**ST**	**Resources (machine, tool and worker)**		**pt (s)**
a_1	Aircoil winding	ST1	Auto CNC Aircoil Machine	W1	5.1
a_2	Aircoil leadout flatenning	ST2	Pneumatic press 1	W2	7.8
a_3	Aircoil leadout trimming	ST3	Pneumatic press 2	W3	6.1
a_4	Aircoil leadout stripping (upper side)	ST4	Stripping machine 1	W4	8.3
a_5	Aircoil leadout side stripping (lower side)	ST4	Stripping machine 2	W5	7.8
a_6	Leads dip soldering	ST5	Solder pot Tweezer Flux	W6	4.5
a_7	Flux cleaning	ST6	Dish washer	W6	0.8
a_8	Aircoil leadout forming	ST7	Pneumatic Forming Machine	W7	4.4
a_9	Rod core assembly to aircoil	ST8	Bent tip tweezer Varnish container	W8	4.6
a_{10}	Rod curing	ST9	Oven Baking tray	W8	4.0
a_{11}	Moulding press	ST10	Double acting compression moulding	W9	8.1
a_{12}	Inductor clamping	ST11	Tongs Clamping machine	W10	3.2

Work element		ST	Resources (machine, tool and worker)		pt (s)
a_{13}	Unit curing	ST12	Oven Baking trolley PC profiler	W10	9.0
a_{14}	Unit unclamping from tongs	ST13	Tongs Clamping machine	W10	1.6
a_{15}	Lead cropping and forming	ST14	Semi-auto crop & form machine	W11	4.7
a_{16}	Part number marking	ST15	Video jet printer	W12	2.2
a_{17}	IR-reflow	ST16	IR-Reflow machine Baking tray	W12	2.3
a_{18}	VMI, Inductor/DCR + Q-factor	ST17	Mantis scope Height Gauge LCR meter	W13	6.4
a_{19}	Packaging	ST17	Tape & reel machine	W13	1.5

Indicator: ST = Workstation, W = Worker, pt = processing time

B. Results and Discussion

An industrial data collection for a selected product which is HM72A-10 series model has been conducted to collect the necessary data such as precedence relations, tasks time and resources used. The current layout of the selected product consist of 19 tasks that were assigned to 17 workstations with a total of 48 resources were used. A simulation of existing layout had been conducted using Witness™ software to simulate the assembly line. Witness™ is a simulation software that commercially used to provide overall view on all the process in terms of busy, idle, blocked and output. The purpose of existing layout simulation is to validate the simulation model with actual layout.

Table II shows the proposed task assignment. It clarifies the details on what task has been assigned in each workstation and their respective total processing time. However after the validation stage, the industrial expert decided that task a_1 (aircoil winding) and task a_2 (aircoil leadout flattening) cannot be assigned in one workstation. Yet, it should be in two different workstations. The highest processing time recorded in Table 2 is 13.1 seconds which is in workstation 4 (ST4).

Table II. PROPOSED TASK ASSIGNMENT

ST	**Task**	**Resources**	**Total processing time (s)**
ST1	a_1 – aircoil winding	W1, W2 Auto CNC aircoil machine Pneumatic press 1	12.9
	a_2 – aircoil leadout flattening		
ST2	a_3 – aircoil leadout trimming	W3 Pneumatic press 2	6.1
ST3	a_4 – aircoil leadout stripping	W4 Stripping machine 1	8.3
WS	**Task**	**Resources**	**Total processing time (s)**
ST4	a_5 - aircoil leadout side stripping	W5 Stripping machine 2 Solder pot Tweezer Flux Dish washer	13.1
	a_6 – leads dip soldering		
	a_7 – flux cleaning		
ST5	a_8 – aircoil leadout forming	W6 Pneumatic forming machine Bent tip tweezer Varnish container Oven Baking tray	13.0
	a_9 – rod core assembly to aircoil		
	a_{10} – rod curing		
ST6	a_{11} – moulding press	W7 Double acting compression moulding Tong Clamping machine	11.3
	a_{12} – inductor clamping		
ST7	a_{13} – unit curing	W8 Oven Baking tray PC profiler	10.6
	a_{14} - unit unclamping from tongs		
ST8	a_{15} – lead cropping and forming	W9 Semi auto cropping and forming machine Video jet printer IR-reflow machine Baking tray	9.2
	a_{16} – part number marking		
	a_{17} – IR-reflow		
ST9	a_{18} – VMI, inductor/DCR + Q-factor	W10 Mantis scope Height gauge LCR meter Tape and reel machine	7.9
	a_{19} - Packaging		

Table III indicates the task assignment after have been validated. The table clearly shows that aircoil winding (a1) and aircoil leadout flattening (a2) are individually assigned in workstation 1 and workstation 2. Therefore, the number of workstation has been increased from 9 workstations to 10 workstations after the validation. The highest processing time is remained unchanged which is 13.1 seconds meanwhile, the lowest processing time is 5.1 seconds.

Table III. TASK ASSIGNMENT AFTER VALIDATION

ST	Task	Resources	Total processing time (s)
ST1	a_1 – aircoil winding	W1 Auto CNC aircoil machine	5.1
ST2	a_2 – aircoil leadout flattening	W2 Pneumatic press 1	7.8
ST3	a_3 – aircoil leadout trimming	W3 Pneumatic press 2	6.1
ST4	a_4 – aircoil leadout stripping (upper side)	W4 Stripping machine 1	8.3
ST	**Task**	**Resources**	**Total processing time (s)**
ST5	a_5 - aircoil leadout side stripping (lower side)	- W5 - Stripping machine 2 - Solder pot - Tweezer - Flux - Dish washer	13.1
	a_6 – leads dip soldering		
	a_7 – flux cleaning		
ST6	a_8 – aircoil leadout forming	- W6 - Pneumatic forming machine - Bent tip tweezer - Varnish container - Oven - Baking tray	13.0
	a_9 – rod core assembly to aircoil		
	a_{10} – rod curing		
ST7	a_{11} – moulding press	-W7 - Double acting compression moulding - Tong - Clamping machine	11.3
	a_{12} – inductor clamping		
ST8	a_{13} – unit curing	-W8 - Oven - Baking tray - PC profiler	10.6
	a_{14} - unit unclamping from tongs		
ST9	a_{15} – lead cropping and forming	-W9 - Semi auto cropping and forming machine - Video jet printer - IR-reflow machine - Baking tray	9.2
	a_{16} – part number marking		
	a_{17} – IR-reflow		
ST10	a_{18} – VMI, inductor/DCR + Q-factor	-W10 - Mantis scope - Height gauge - LCR meter - Tape and reel machine	7.9
	a_{19} – Packaging		

Table IV shows the simulation results of existing layout, after optimization using NSGA-II and the result after validation. The validation is conducted by an interview and discussion session to determine either the optimization result using the proposed method is acceptable or not. For the validation purpose, some queries has been raised during the interview and discussion session; (i) Do the proposed layout is possible to be implemented in the production line? (ii) Do the effectiveness of the line achieved the industrial target?

The most striking observation to emerge from the results of comparison was the number of workstations are extensively decreased after the NSGA-II optimization from 17 workstations that were used for the existing layout to 9 workstations. The rapid decrease in the number of workstation is because of all the tasks that used same type of resources will be assigned to one workstation subject to the constraints i.e. (i) the precedence relations are not violated (ii) total processing time in each workstation does not exceed the cycle time.

However, the number of workstations has been increased to 10 after the validation. This is due to some of the tasks cannot be assigned to the same workstation. This situation caused the value of busy percentage in workstation after the validation turn out to be less (70.5%) compared with after the optimisation (78.3%). However, both values remain lower compare to the busy percent of workstation of the existing layout which is 33.7%. In fact, the number of resources being used also show a reduction of 3 resources both after the optimization and validation.

The efficiency of the line is calculated using (1) as follows:

$$E = \frac{\sum_{i=1}^{m} T_i}{mc} \times 100\% \qquad (1)$$

where E: Line efficiency

m: Number of workstation

c: Cycle time

T_i: Total processing time of the i^{th} workstation

The simulation results indicates that the line efficiency of the existing layout is the worst among the three results i.e. existing layout, 33.8%; after optimization, 78.4%; after validation, 70.5%. This can be concluded that the most efficient line was after the optimization. Meanwhile, the percentage value of blocked in workstation is the lowest after the optimization (9.7%), compared to the result after the validation and existing layout which is 18.8% and 10.5% correspondingly. Besides, the results show that the percentage busy of worker after the optimization is the same as after the validation which is 70.5%. This is due to the reason of the number of worker assigned to all workstations in the both phases are the same. The idle percentage of worker for both stages are also comparable which is 29.5%.

Table IV. COMPARISON OF EXISTING, NSGA-II OPTIMIZATION AND VALIDATION RESULTS

Data	Existing	After NSGA-II optimization	After validation
NWS	17	9	10
CT	16.1	13.1	13.1
Resource	43	40	40
% Line eff.	33.8	78.4	70.5
%Busy (Workstation)	33.7	78.3	70.5
%Idle (Workstation)	55.7	12.0	10.7
%Blocked (Workstation)	10.5	9.7	18.8
%Busy (Worker)	44.1	70.5	70.5
%Idle (Worker)	55.9	29.5	29.5
Daily output	4914	6039	6039

In the meantime, the existing layout shows the worst reading for busy percentage of worker (44.1%) and also the percentage of idle of worker (55.9%). The results obtained from the NGSA-II optimization shows that the proposed method can be implemented in manufacturing industry for the target to enhance the industrial productivity as well as increase the line efficiency. The validation from industrial expert concluded that the proposed layout was a worthy plan as it can minimise the number of resources used and number of workstations. On top of that, the efficiency and the productivity of the line also can be increased.

As we can see from Table IV, the daily output obtained from the existing layout is 4914 units, while the output achieved after both the optimisation and validation is 6039 units per day. Apart from the optimisation parameters, the output of the production was increased as well. Thus, the proposed method and the optimisation algorithm are applicable for industrial application.

CONCLUSION

This paper presents a case study to optimize assembly line balancing type-e problem with resource constraint (ALBE-RC) by using elitist non-dominated sorting genetic algorithm (NSGA-II). The finding from the industrial case study provides evidence that the results of optimization have improvement in term of line efficiency, daily output, number of workstations, cycle time and also the usage of resources compared with the existing layout. The validation from the industrial expert also shows that the proposed method is applicable and can be implemented for industrial application.

ACKNOWLEDGMENT

We acknowledge the Ministry of Higher Education for funding this study through a research grant under RDU140103. The authors would like to thank Universiti Malaysia Pahang for providing the necessary facilities.

REFERENCES

[1] M. F. F. Rashid, W. Hutabarat, and A. Tiwari, "A review on assembly sequence planning and assembly line balancing optimisation using soft computing approaches," *The International Journal of Advanced Manufacturing Technology,* vol. 59, pp. 335-349, 2012.

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

[3] Z. RuiJun, C. DingFang, W. Yong, Y. ZhongHua, and W. Xinxin, "Study on line balancing problem based on improved genetic algorithms," in *Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on*, 2007, pp. 2033-2036.

[4] W. Zhang, M. Gen, and L. Lin, "A multiobjective genetic algorithm for assembly line balancing problem with worker allocation," in *Systems, Man and Cybernetics, 2008. SMC 2008. IEEE International Conference on*, 2008, pp. 3026-3033.

[5] N. Hamta, S. M. T. Fatemi Ghomi, F. Jolai, and M. Akbarpour Shirazi, "A hybrid PSO algorithm for a multi-objective assembly line balancing problem with flexible operation times, sequence-dependent setup times and learning effect," *International Journal of Production Economics,* vol. 141, pp. 99-111, 2013.

[6] O. Battaïa and A. Dolgui, "A taxonomy of line balancing problems and their solutionapproaches," *International Journal of Production Economics,* vol. 142, pp. 259-277, 2013.

[7] K. Ağpak and H. Gökçen, "Assembly line balancing: Two resource constrained cases," *International Journal of Production Economics,* vol. 96, pp. 129-140, 2005.

[8] A. Corominas, L. Ferrer, and R. Pastor, "Assembly line balancing: general resource-constrained case," *International Journal of Production Research,* vol. 49, pp. 3527-3542, 2011.

[9] T. R. Browning and A. A. Yassine, "A random generator of resource-constrained multi-project network problems," *Journal of scheduling,* vol. 13, pp. 143-161, 2010.

[10] T. Al-Hawari, M. Ali, O. Al-Araidah, and A. Mumani, "Development of a genetic algorithm for multi-objective assembly line balancing using multiple

assignment approach," *The International Journal of Advanced Manufacturing Technology,* pp. 1-14, 2014.

[11] P. T. Zacharia and A. C. Nearchou, "A meta-heuristic algorithm for the fuzzy assembly line balancing type-E problem," *Computers & Operations Research,* vol. 40, pp. 3033-3044, 2013.

[12] E. Gurevsky, O. Battaïa, and A. Dolgui, "Balancing of simple assembly lines under variations of task processing times," *Annals of Operations Research,* vol. 201, pp. 265-286, 2012.

[13] M. Jusop and M. Ab Rashid, "A review on simple assembly line balancing type-e problem," in *IOP Conference Series: Materials Science and Engineering*, 2015, p. 012005.

[14] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," *Lecture notes in computer science,* vol. 1917, pp. 849-858, 2000.

[15] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *Evolutionary Computation, IEEE Transactions on,* vol. 6, pp. 182-197, 2002.

[16] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*, 1st ed.: John Wiley & Sons, Ltd, 2001.

[17] R. Saravanan, *Manufacturing Optimization through Intelligent Techniques*: Taylor & Francis Group, 2006.

[18] M. Alvares Barbosa Junior, F. B. de Lima Neto, and T. Marwala, "Optimizing risk management using NSGA-II," in *Evolutionary Computation (CEC), 2012 IEEE Congress on*, 2012, pp. 1-8.

[19] A. Baykasoğlu and L. Özbakır, "Discovering task assignment rules for assembly line balancing via genetic programming," *The International Journal of Advanced Manufacturing Technology,* vol. 76, pp. 417-434, 2014.

Assembly Line Balancing with Resource Constraints using New Rank-Based Crossovers

N.H. Kamarudin, M.F.F. Ab. Rashid

Faculty of Mechanical Engineering, Universiti Malaysia Pahang, 26600 Pekan, Malaysia.

Email: ffaisae@ump.edu.my

Abstract. Assembly line balancing (ALB) is about distributing the assembly tasks into workstations with the almost equal workload. Recently, researchers started to consider the resource constraints in ALB such as machine and worker, to make the assembly layout more efficient. This paper presents an ALB with resource constraints (ALB-RC) to minimize the workstation, machine and worker. For the optimization purpose, genetic algorithm (GA) with two new crossovers is introduced. The crossovers are developed using ranking approach and known as rank-based crossover type I and type II (RBC-I and RBC-II). These crossovers are tested against popular combinatorial crossovers using 17 benchmark problems. The computational experiment results indicated that the RBC-II has better overall performance because of the balance between divergence and guidance in the reproduction process. In future, the RBC-I and RBC-II will be tested for different variant of ALB problems.

1. Introduction

Assembly line balancing (ALB) plays a vital function in a production system. The installation of an assembly line is a long-term decision and requires large capital investments. It is important that such a system is designed and balanced so that it works as efficiently as possible [1]. The simplest version of ALB problem is known as simple assembly line balancing problem (SALBP) which also known as One-sided ALB [2]. SALBP deals with a serial assembly line which processes a unique model of a single product. In previous research, a lot of attention has been given to this type of problem.

However, in the majority of the previous works, researchers make assumptions where any of assembly tasks can be processed or assembled in any workstations. This is certainly true for the product which only requires a common or simple tool to be assembled. However, when the complexity of a product increased, it requires a special tool, machine or highly skilled labor to assemble that particular component. Therefore, the limitation of resources will be another constraint for the industry. In fact, the issue of line balancing with the minimum number of resources has always been a serious problem in the industry [3]. This problem is known as assembly line balancing with resource constraints (ALB-RC)

Previously, researchers had studied the line balancing with resource constraints. [4] started the ALB-RC by considering two resources and solve the problem using integer programming. Next, [5] proposed a model to support generalized constraints problem. [6] later on model and optimize the ALB with worker skill constraint. The purpose is to match the assembly task with the level of the worker skill. Besides that, [7] optimize the multi-objective ALB with general resources using domination concept.

Researchers also implemented different algorithms to optimize the ALB-RC problem. [8] combined priority rule-based method (PRBM) and genetic algorithm (GA) to optimize this problem. The PRBM is used to generate initial chromosomes for GA. Meanwhile, [9] implement the hybrid multi-objective genetic algorithm (MOGA) to optimize the problem to obtain Pareto front. In addition, [10] implemented elitist non-dominated sorting GA (NSGA-II) to optimize this problem.
This paper extends the existing ALB-RC by considering the worker selection, besides the workstation number and tool resource. In this problem, the engineer has a different option for workers with different ability to conduct assembly task. The problem is later optimized by GA with new crossover operators.

2. ALB-RC Problem Modelling

The ALB problem is represented using a precedence graph. The number inside the node represents the assembly task. The directed edge means the precedence between task *i* and *j*. In ALB, the assembly tasks need to be assigned into workstations, so that the workstation time is almost equal. To presents the ALB-RC, the following example is used. Figure 1 shows a precedence graph that represents an assembly process. Each of nodes represents the assembly task while the arrows represent the precedence.

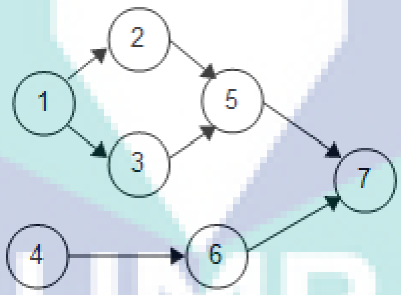


Figure 1. Example of precedence graph

Table 1 meanwhile shows the assembly information which includes the task time, tool and also worker. The worker columns with tick mark meaning that the worker is able to conduct a specific assembly task. To assemble an assembly task, only one worker is required.

Table 1. Assembly information for Figure 1

Task	Time	Tool	Workers						
			1	2	3	4	5	6	7
1	18	A	/		/	/		/	/
2	22	B		/		/		/	
3	9	B	/	/	/		/		/
4	7	A			/		/	/	/
5	12	A	/			/		/	
6	6	B		/	/		/		
7	20	A	/	/		/			/

For clarity of the ALB-RC evaluation, let consider a feasible assembly sequence f_1 = [1 4 3 2 6 5 7]. For this example, the maximum cycle time, ct_{max} is 34 time unit. It means that the workstation time cannot exceed the ct_{max} or otherwise, the demand for the product cannot be fulfilled. In Table 2, the Worker row shows the entire workers that capable to conduct a specific assembly task.

Table 2. Example of a feasible assembly sequence

Sequence	1	4	3	2	6	5	7
Time	18	7	9	22	6	12	20
Tool	A	A	B	B	B	A	A
Worker	1,3,4,6,7	3,5,6,7	1,2,3,5,7	2,4,6	2,3,5	1,4,6	1,2,4,7

The assembly tasks were assigned into the workstation as shown in Table 3. The station time row shows cumulative time to conduct assembly process for all tasks in a specific station. The Tool row shows the required tool to conduct assembly process in a specific station. Meanwhile, the worker selection is made based on the number of workers frequency in a workstation. For example, in workstation 1, workers 3 and 7 have the highest frequency. In this case, the worker is select randomly.

Table 3. Assembly task and workstation assignment

Workstation	1	2	3
Task	1, 4, 3	2, 6	5, 7
Station time	34	28	32
Tool	A,B	B	A
Worker	3	2	1

Based on the presented approach, the objective function can be measured as follow:

Number of workstation = 3

Number of tool = 4

Number of worker = 3

3. Genetic Algorithm

Genetic algorithm is an optimization technique that mimics the survival for the fitness concept. Solutions with better fitness have larger possibilities to remain in the population, while the solution with bad fitness will be eliminated from the population [11]. In general, GA consists of five main steps; Initialization, Evaluation, Selection, Crossover and Mutation. The algorithm is coded using permutation number to represent the assembly tasks. However, due to the randomness of permutation, the generated number may violate the precedence relation in assembly. Therefore, topological sort based on the earliest task appearance is implemented to decode the solution.

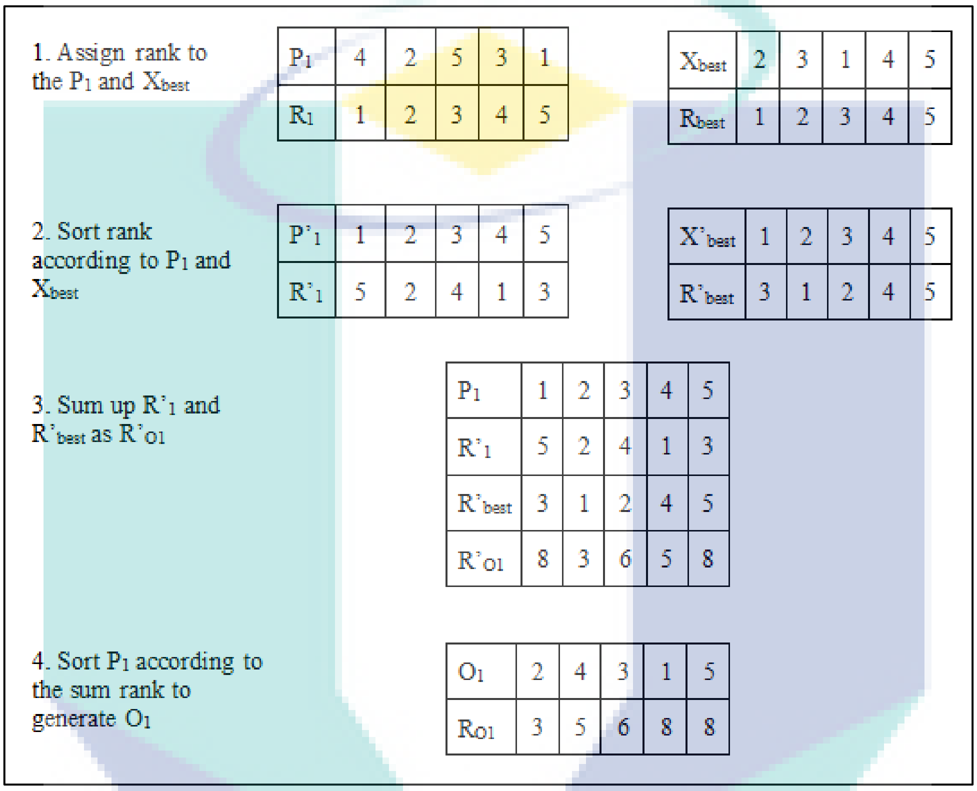
The purpose of selection step is to choose the chromosome to be placed in the mating pool. The selected chromosome will be the parent of the children in a new generation. The selection process is conducted using Roulette wheel selection (RWS) mechanism. Meanwhile, for the crossover, we introduced two crossover operators, named Rank based crossover type I and II (RBC-I and RBC-II). The proposed crossovers are compared with popular crossover operator for permutation problem, i.e. ordered crossover (OX), partially matched crossover (PMX) and Moon crossover [12].

3.1. Proposed Rank Based Crossover

Both of the proposed crossovers taken into account the best chromosome from the population in the reproduction process. In both of the proposed crossover, each of the assembly tasks will be given a rank according to their position in the chromosome. Then the rank for parent and best chromosome will be summed up to form a new rank. The child solution will be generated based on the new rank. By using this approach, the new child will inherit the gene from their parent and also the best solution.

3.1.1. Rank based crossover type-I (RBC-I)

In RBC-I, the parent rank (R_1) and the best solution rank (R_{best}) will be added (Figure 2). Next, the rank is sorted according to the parent (P_1) and the best solution (X_{best}). The sorted parent rank (R'_1) and the best rank (R'_{best}) is added to form sorted offspring rank (R'_{O1}). Finally, the P'_1 is sorted according to the R'_{O1} to generate offspring solution, O_1. In the case where the rank is tied, the selection is made randomly. The numerical procedure for RBC-I is presented in Figure 2.



1. Assign rank to the P_1 and X_{best}

P_1	4	2	5	3	1
R_1	1	2	3	4	5

X_{best}	2	3	1	4	5
R_{best}	1	2	3	4	5

2. Sort rank according to P_1 and X_{best}

P'_1	1	2	3	4	5
R'_1	5	2	4	1	3

X'_{best}	1	2	3	4	5
R'_{best}	3	1	2	4	5

3. Sum up R'_1 and R'_{best} as R'_{O1}

P_1	1	2	3	4	5
R'_1	5	2	4	1	3
R'_{best}	3	1	2	4	5
R'_{O1}	8	3	6	5	8

4. Sort P_1 according to the sum rank to generate O_1

O_1	2	4	3	1	5
R_{O1}	3	5	6	8	8

Figure 2. Numerical procedure for RBC-I

3.1.2. Rank based crossover type-II (RBC-II)

The RBC-II applied the same rank concept as in RBC-I, but this crossover considers two parents. The early steps where the rank is assigned and sorted is the same with RBC-I as shown in Figure 3. To calculate the rank for offspring solutions (R'_O), the following formula is used in RBC-II.

$$R'_O = C_{best}\,(R'_{best}) + C_1(R'_1) + C_2(R'_2) \qquad (1)$$

C_{best}, C_1 and C_2 are the coefficients for the X_{best}, P_1 and P_2 respectively. The C_{best} is fixed at 0.2. Meanwhile, the C_1 and C_2 coefficient is depend on the offspring. To generate offspring 1 (O_1), the C_1 and C_2 are as follow.

$$C_1 = 0.7(1 - C_{best}) \qquad (2)$$

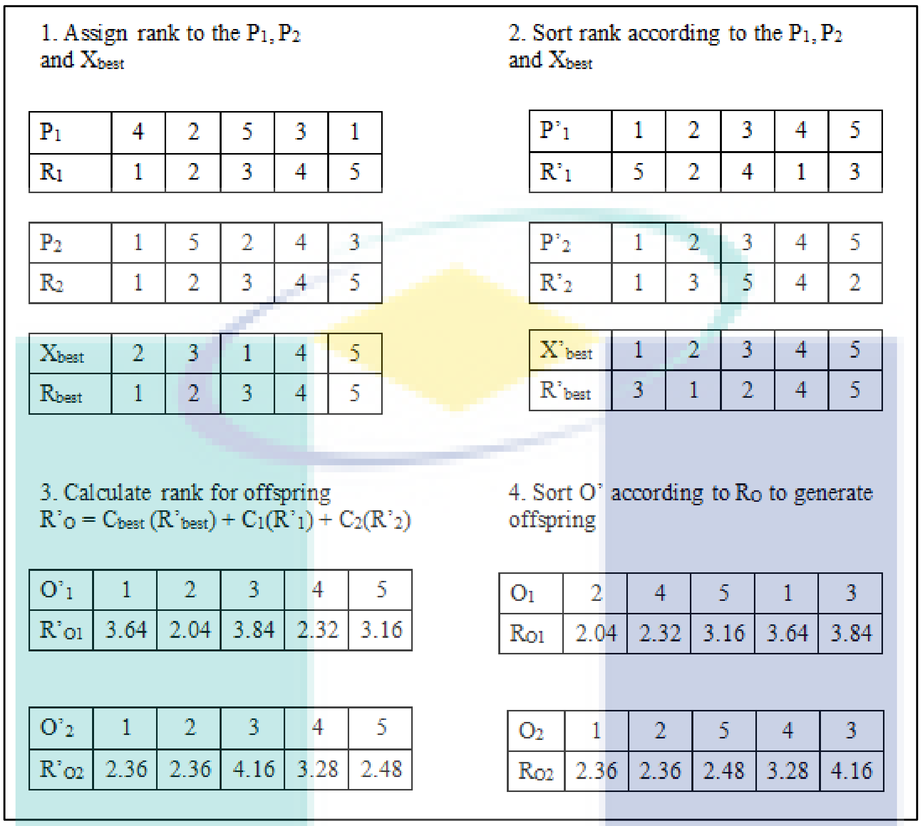
$$C_2 = 0.3(1 - C_{best}) \qquad (3)$$

On the other hand, to generate offspring 2 (O_2), the following coefficients are used.

$$C_1 = 0.3(1 - C_{best}) \qquad (4)$$

$$C_2 = 0.7(1 - C_{best}) \qquad (5)$$

The offspring solutions (O_1 and O_2) are generated by sorting the R'_O in the ascending orders. As in RBC-I, in the event of tie rank, the selection is made randomly. The numerical example for RBC-II is shown in Figure 3.



1. Assign rank to the P_1, P_2 and X_{best}

P_1	4	2	5	3	1
R_1	1	2	3	4	5

P_2	1	5	2	4	3
R_2	1	2	3	4	5

X_{best}	2	3	1	4	5
R_{best}	1	2	3	4	5

2. Sort rank according to the P_1, P_2 and X_{best}

P'_1	1	2	3	4	5
R'_1	5	2	4	1	3

P'_2	1	2	3	4	5
R'_2	1	3	5	4	2

X'_{best}	1	2	3	4	5
R'_{best}	3	1	2	4	5

3. Calculate rank for offspring
$R'_O = C_{best}(R'_{best}) + C_1(R'_1) + C_2(R'_2)$

O'_1	1	2	3	4	5
R'_{O1}	3.64	2.04	3.84	2.32	3.16

O'_2	1	2	3	4	5
R'_{O2}	2.36	2.36	4.16	3.28	2.48

4. Sort O' according to R_O to generate offspring

O_1	2	4	5	1	3
R_{O1}	2.04	2.32	3.16	3.64	3.84

O_2	1	2	5	4	3
R_{O2}	2.36	2.36	2.48	3.28	4.16

Figure 3. Numerical example of RBC-II

4. Computational Experiment

A computational experiment has been conducted to measure the performance of RBC-I and RBC-II. For this purpose, a set of ALB benchmark problem by Scholl is used [13]. The benchmark set consist of 17 problems that varies in term of the size. The benchmark test problem is divided into three categories; small ($n \leq 20$ task), medium ($20 < n \leq 70$) and large ($n > 70$). For comparison purpose, the RBC-I and RBC-II are compared with popular crossover operators for the combinatorial problem. The comparison crossovers are the ordered crossover (OX), partially matched crossover (PMX) and Moon crossover. The OX and PMX are among popular crossover operator for the combinatorial problem. Meanwhile, the Moon crossover is used since it was claimed to be the best crossover for the combinatorial problem [12]. For the computational experiment, the population size is set to 30, maximum generation is 300, probability of crossover is 0.7 and probability of mutation is 0.2. The optimization is run for ten times to reduce the pseudorandom effect. Table 4 presents the best fitness obtained by GA using different crossover strategies.

Based on the results in Table 4, all the crossovers were able to search for an optimum solution for the small size problems. However, when the problem size is increased, the RBC-I and RBC-II has better performance compared with other crossovers, except in Hahn problem. In medium size problem, the RBC-II has better fitness in 83% of the problem. Meanwhile, in large size problem, the RBC-I and II individually has better fitness in 50% of the problem.

To have better view from the computational experiment result, a standard competition ranking approach is used. The crossover with the best fitness will be given rank 1, followed by the next as rank 2 etc. If the crossover performance is equivalent, the following rank is ignored. The summary of the standard competition ranking is presented in Table 5.

Table 4. Optimization results

No	Problem	No. of Task	Given Cycle Time	Crossover type				
				OX	PMX	Moon	RBC-I	RBC-II
1	Mertens	7	8	**8.7500**	**8.7500**	**8.7500**	**8.7500**	**8.7500**
2	Bowman	8	20	**8.5000**	**8.5000**	**8.5000**	**8.5000**	**8.5000**
3	Jaeschke	9	18	**3.5000**	**3.5000**	**3.5000**	**3.5000**	**3.5000**
4	Mansoor	11	48	**2.9286**	**2.9286**	**2.9286**	**2.9286**	**2.9286**
5	Jackson	11	13	**2.2857**	**2.2857**	**2.2857**	**2.2857**	**2.2857**
6	Buxey	29	54	4.0457	4.0576	4.0517	3.7789	**3.5181**
7	Sawyer	30	75	1.6800	1.6800	1.6800	1.8000	**1.4400**
8	Gunther	35	69	2.7952	2.8952	2.7810	2.6738	**2.5881**
9	Kilbridge	45	69	3.8831	3.9642	3.9599	3.8789	**3.7999**
10	Hahn	53	2004	5.6467	**5.5190**	5.5815	5.6495	5.6440
11	Warnecke	58	111	3.4024	3.5080	3.8168	3.5628	**3.1858**
12	Wee Mag	75	56	4.1857	4.1303	4.0446	4.0053	**3.8964**
13	Arc83	83	6540	2.4743	2.5198	2.5879	**2.4320**	2.5132
14	Lutz 2	89	19	3.0760	2.9237	3.0062	**2.5626**	2.9492
15	Mukherje	94	263	3.2866	3.3370	3.2747	3.4274	**3.0903**
16	Arc111	111	6540	6.7874	6.6003	6.5334	6.4993	**5.5394**
17	Barthol2	148	170	3.2254	3.1278	3.1912	**2.9669**	3.1034

Table 5. Summary of standard competition ranking

Crossover	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5	Average rank
OX	5	3	3	2	4	2.8235
PMX	6	2	2	4	3	2.7647
Moon	5	3	3	4	2	2.7058
RBC-I	8	5	0	1	3	2.1764
RBC-II	13	1	3	0	0	1.4117

Based on Table 5, the RBC-II was most frequently ranked as 1, followed by RBC-I and PMX. Meanwhile, the OX has the most frequently ranked as 5. The average rank for each of crossover is then calculated. Based on the average rank the best crossover is RBC-II. The RBC-II is only ranked from 1 until 3. In the meantime, RBC-I is in the second best according to the average rank. For RBC-I, besides ranked as 1 and 2, this crossover was also ranked as 5 in three cases. On the other hand, the OX is the worst crossover based on the average rank.

The RBC-I and II have shown better performance because of the involvement of the best chromosome in the reproduction process. This makes the search direction is more guided compared with other crossovers. In the OX, PMX and Moon crossovers, the reproduction process solely depend on the parents. Even though the parents were selected among the best, the variation in the chromosomes makes the search direction become too diverse.

Meanwhile, in the comparison between RBC-I and II, the RBC-I is too dependent on the best solution because a single parent is mated with the best solution for the regeneration. This makes the chance for the chromosome to trap in local optima is slightly higher. In RBC-II, the regeneration process involved a pair of parents and the best solution. Two chromosomes from parents make the regeneration is not too relied on the best solution. Furthermore, the generated offspring only inherit 20% of the gene from the best solution (since C_{best} = 0.2). This makes the RBC-II able to generate more varied offspring, but in the guided mode.

5. Conclusions

This paper presents an assembly line balancing with resource constraints. In particular, besides balancing the assembly workload in the station, this work also consider to minimize the number of machines and workers in an assembly line. For optimization purpose, two crossover operators for genetic algorithm were introduced. The proposed crossovers were based on the assembly sequence rank, known as Rank-based crossover type I and II (RBC-I and RBC-II). In different with other crossover operators, the RBC-I and II consider the best chromosome in the regeneration process.

The computational experiments using 17 benchmark problems indicated that the RBC-II has better overall performance compared with comparison crossovers in the genetic algorithm. The RBC-II performance is because of the balance between divergence and guidance during the reproduction process in the crossover. In future, an industrial case study will be conducted to validate the problem modeling and the RBC-II performance.

Acknowledgments

This research is supported by Ministry of Higher Education Malaysia and Universiti Malaysia Pahang under grant number RDU140103.

6. References

[1] C. Becker and A. Scholl, “A survey on problems and methods in generalized assembly line balancing,” *Eur. J. Oper. Res.*, vol. 168, no. 3, pp. 694–715, Feb. 2006.

[2] K. Ağpak and S. Zolfaghari, “Mathematical models for parallel two-sided assembly line balancing problems and extensions,” *Int. J. Prod. Res.*, pp. 1–13, Sep. 2014.

[3] N. Hamta, S. M. T. Fatemi Ghomi, F. Jolai, and M. Akbarpour Shirazi, “A hybrid PSO algorithm for a multi-objective assembly line balancing problem with flexible operation times, sequence-dependent setup times and learning effect,” *Int. J. Prod. Econ.*, vol. 141, no. 1, pp. 99–111, Jan. 2013.

[4] K. Ağpak, H. Gökçen, K. Ağpak, and H. Gökçen, “Assembly line balancing: Two resource constrained cases,” *Int. J. Prod. Econ.*, vol. 96, no. 1, pp. 129–140, Apr. 2005.

[5] A. Corominas, L. Ferrer, and R. Pastor, “Assembly line balancing: General resource-constrained case,” *Int. J. Prod. Res.*, vol. 49, no. 12, pp. 3527–3542, Jun. 2011.

[6] T. Koltai and V. Tatay, “Formulation of workforce skill constraints in assembly line balancing models,” *Optim. Eng.*, vol. 14, no. 4, pp. 529–545, 2013.

[7] M. Jusop and M. F. F. A. Rashid, “Optimisation of assembly line balancing type-E with resource constraints using NSGA-II,” *Key Eng. Mater.*, vol. 701, pp. 195–199, 2016.

[8] N. T. P. Quyen, J. C. Chen, and C.-L. Yang, “Hybrid genetic algorithm to solve resource constrained assembly line balancing problem in footwear manufacturing,” *Soft Comput.*, pp. 1–17, 2016.

[9] H. Triki, A. Mellouli, and F. Masmoudi, “A multi-objective genetic algorithm for assembly line resource assignment and balancing problem of type 2 (ALRABP-2),” *J. Intell. Manuf.*, vol. 28, no. 2, pp. 371–385, 2017.

[10] M. Jusop and M. Rashid, “Optimization of Assembly Line Balancing with Resource Constraint using NSGA-II : A Case Study,” *Int. J. Appl. Eng. Res.*, vol. 12, no. 7, pp. 1421–1426, 2017.

[11] Y. Delice, E. Kızılkaya Aydoğan, and U. Özcan, “Stochastic two-sided U-type assembly line balancing: a genetic algorithm approach,” *Int. J. Prod. Res.*, vol. 54, no. 11, pp. 3429–3451, 2016.

[12] C. Moon, J. Kim, G. Choi, and Y. Seo, “An efficient genetic algorithm for the traveling salesman problem with precedence constraints,” *Eur. J. Oper. Res.*, vol. 140, no. 3, pp. 606–617, Aug. 2002.

[13] A. Scholl, “Benchmark Data Sets by Scholl,” *Assembly Line Balancing Data Dets & Research Topics*, 1993. [Online]. Available: http://assembly-line-balancing.mansci.de/salbp/benchmark-data-sets-1993/.

Optimization of Mixed-Model Assembly Line Balancing Problem with Resource Constraints

Muhamad Magffierah Razali
*Mohd Fadzil Faisae Ab.Rashid**
Muhammad Razif Abdullah Make
Faculty of Mechanical Engineering
Universiti Malaysia Pahang
26600 Pekan, Pahang, Malaysia
*ffaisae@ump.edu.my**

ABSTRACT

In this study, mixed-model assembly line balancing problem (MMALBP) is optimized using four Meta-heuristic Algorithms (MAs), namely particle swarm optimization (PSO), simulated annealing (SA), ant colony optimization (ACO) and genetic algorithm (GA). Three categories of test problem (small, medium, and large) is used ranging from 8 to 100 number of tasks. For computational experiment, MATLAB software is used in investigate the MAs performances to optimize the designated objective functions. The results reveal that ACO algorithm performed better in term of finding the best fitness functions when dealing with a large number of tasks. Averagely, it has produces better solution quality than PSO by 5.82%, GA by 9.80%, and SA by 7.66%. However, PSO more superior in term of processing time compared to ACO by 29.25%, GA by 40.54%, and SA by 73.23%. Hence, future research directions such as using the actual manufacturing assembly line data to test the algorithm performances are likely to happen.

Keywords: *Mixed-model Assembly Line Balancing; Meta-heuristic algorithm; Resource constraints, Optimization Algorithm*

Introduction

Assembly line balancing problem (ALBP) is a matter of decisions that arise when designing or redesigning the assembly line and it involves finding the optimum assignment of tasks. In the recent decades, ALBP has been one of the major interesting research subjects due to its importance to the manufacturers nowadays. This subject is important because manufacturers able to increase the efficiency, productivity as well as gain profit and reduce operational cost by applying assembly line balancing [1]. Mixed-model assembly line balancing problem (MMALBP) is categorized under ALBP. It differs with the other problem in ALBP classification because it deals with an assembly line that capable to assemble more than one different model of product at the same time on the same assembly line [2].

In context of MMALBP, various other factors are considered. As an example, the number of models which will be assembled and total demand throughout planning horizon. There are a few methods that have been used by previous researcher to propose an effective solution for this problem. For instance, researcher utilize a mathematical approach namely mixed integer linear programming and mathematical programming techniques that aimed to optimized MMALBP [3]. However, problems associated with the use of optimization in large-scale problems frequently reach local optimum especially when facing NP-hard problems. The NP-hard problem contains massive number of variables as well as non-linear objective functions make it complicated for the conventional method to deliver a decent solution [4].

In order to counter this problem, some alternative solutions are proposed. Researchers have presented meta-heuristic algorithms to find near-optimal solution to the MMALBP. Meta-heuristic algorithms (MAs) are stochastic optimizer programming that capable to solve multi-objective optimization problems. They able to deal with the multi-objective problems with a set of possible solutions simultaneously [5]. The algorithms can find the near-optimal solution in a single run compared with traditional techniques that need to be executed in a series of separate runs.

MAs are a method that imitate the metaphor of natural biological evolution and the social behavior of species [6]. As an example to that metaphor, in order to find a source of food, ants search the shortest route that can lead to it and how the flock of birds work together to get to destination during their migration. However, the first reported MAs in the previous literature was the genetic algorithm inspired from the Darwin's principle which is based on natural evolution back in 1970s [7]. Same as the metaphor mentioned before, simulated annealing optimization technique is to imitate the physical process of annealing also known as heat treatment process whereby a metal is heated to a specific temperature and then left to cool slowly [8].

In the interest to mimic the behavior of this species effectively, various researchers have developed a computerized system that capable to find solutions for complex optimization problems encourage by aforementioned natural biological evolution and the social behavior of species such as ant colony optimization and particle swarm optimization. Then, measuring the performance of these MAs has been given a wide attention from researchers generally to verify the applicability of a particular algorithm to that particular ALBP. The objectives function is used as the evaluation criteria to measure the performance of a meta-heuristic algorithm that studied. Most common objectives function being studied in the existing literature such as minimize cycle time [9], minimize number of workstation [10], minimize total idle cost [11], balancing the workload [12] and etc.

Despite all the prior effort by the researchers, there was some limitation on the existing works. This include the assumption that all workstation has the same capability in term of resources such as tool, manpower, machine and etc. Furthermore, lack of consideration on existing resources on the manufacturing line and not to mention, recent studies only considered on specific resource constraint in their research.

This paper investigates the mixed-model assembly line balancing problem (MMALBP) with resource constraints. Then, a computational study to compare the performance of four meta-heuristic algorithms in term of fitness value, processing time and quality of the solutions was conducted. These four MAs technique inspired by different natural process namely ant colony optimization (ACO), genetic algorithm (GA), particle swarm optimization (PSO), and simulated annealing (SA). Three objectives function are chosen which is minimize total cycle time, minimize product rate variation, and minimize number of resources used on the assembly line. The selection of these three objective functions were based on literature in MMALBP where it is the most studied by previous researcher.

Problem Modelling

In order to evaluate the performance of selected MAs, a problem modelling was constructed with the objective functions to minimize cycle time, minimize product rate variation (PRV) and resources used on the assembly line [9]. Mixed-model assembly line consists more than one product which will be assembled on the same line. Each model has its own precedence relation diagram that shows an arrangement of task which needed to be completed to produce final finished product. A joint precedence diagram is formed from the combination of two or more product models. A simple illustration on how the join precedence is formed presented in Figure 1.

Each model has six tasks which needed to be completed but may be different in term of the task arrangement. For example, in Model 1, task 2 and 3 must be finished first before able to proceed with task 5 meanwhile in Model 3 only task 2 is needed to be completed before moving to task 5. Then, **Error! Reference source not found.** shows how the joint precedence diagram is formed based on these three models. It shows the proposed sequence of the task that must be followed to assemble the products from start until the final product. All known solutions for MMALBP are relied on the joint precedence diagram which is crucial for solving such problems [13].

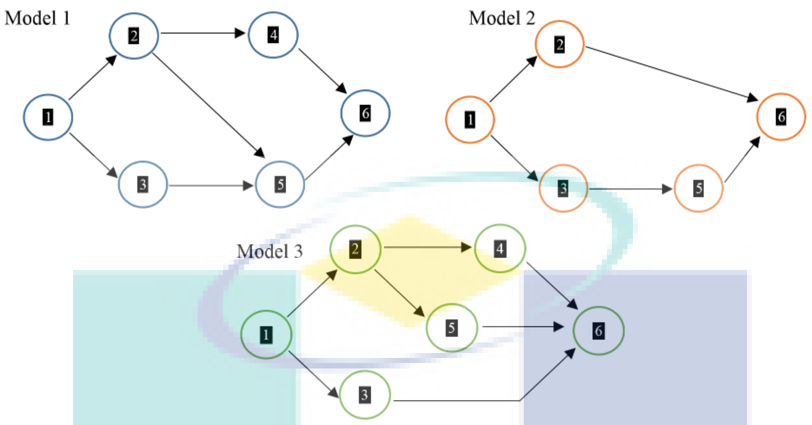


Figure 1: Precedence diagram for model 1, 2, and 3

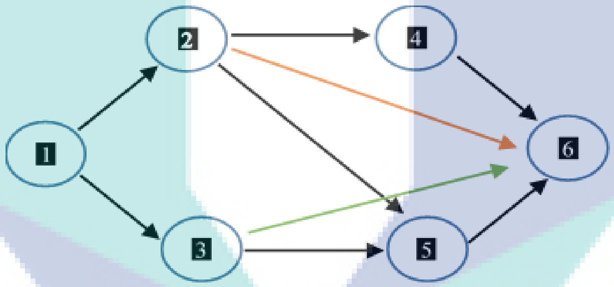


Figure 2: Join precedence diagram

The parameters and indices of the model will be as:

Notation	Definition
S	number of workstations(fixed) $s = 1, 2\ldots, S$
J	number of product models to be assembled $j = 1, 2\ldots, J$
Ne	number of task $e = 1, 2\ldots, Ne$
pre_i	predecessor for task i based on precedence diagram
t_i	execution time for task i
D_T	total quantity of units or total demand
d_j	demand for product j, $j = 1, 2, \ldots, a$
$X_{i,k}$	total quantity of product/produced over stages 1 to k, $k = 1, 2, \ldots, Dt$
$maxR$	maximum resources $r = 1, 2,\ldots, maxR$

C_T	cycle time
T_{ej}	shift task model time
T_e	shift task time
t_e	task time
N_j	demand schedule for each model
U	production rates variation of production sequence

Decision variables

U_{ej}	1 if task, *e* is used on model *j* ; 0,otherwise
X_{es}	1 if task, *e* is assigned to workstation *s*; 0, otherwise
Y_{rs}	1 if resource, *r* is used in workstation *s*; 0,otheriwse

Objectives function and constraints

In this paper, three objective functions are used as the evaluation criteria. The first one is minimizing the cycle time. Second is to minimize product rate variation (PRV) and third is to minimize resources used on the assembly line. These selected objectives function and its related constraint are formulated as below [9]:

$$f1 = min \sum_{e=1}^{N_e} \sum_{j=1}^{J} C_T \tag{1}$$

$$f2 = min \sum_{s=1}^{S} \sum_{r=1}^{max_R} Y_{rs} \tag{2}$$

$$f3 = \min \sum_{k=1}^{D_T} \sum_{j=1}^{J} \left(x_{j,k} - k \times \frac{d_j}{D_T} \right)^2 \tag{3}$$

Objective function *f1*, in equation (1) is to minimize cycle time meanwhile *f2*, in equation (2) aim to minimize resources used on assembly line and *f3* in equation (3) is to minimize product rate variation (PRV) based on the demand of planning horizon. These three objective functions are bound by these following restriction:

$$\sum_{s=1}^{S} X_{as} - \sum_{s=1}^{S} X_{bs} \leq 0, for\ \forall (a,b) \in pre_i \tag{4}$$

$$\sum_{i\in wk} t_i(X_{es}) \leq C, \qquad s = 1,\dots,S \tag{5}$$

$$\sum_{s=1}^{S} X_{es} = 1, \qquad e = 1,\dots,n \tag{6}$$

Constraint (4) is to assure the precedence constraints among the tasks is followed, which is to guarantees that no successor task is appointed to an earlier station. Meanwhile, inequality (5) is to ensure that total task times assigned to each station does not surpass the designated maximum cycle time. Restriction of each task can only be assigned to one workstation is created by using constraint (6). The maximum cycle time mentioned in this paper is stated as reference cycle time, Ref_{CT} and can be expressed as:

$$Ref_{CT} = \frac{\sum shift\ task\ time, T_e}{no.\ of\ workstation, s}, \quad s = 1,\dots,S \tag{7}$$

A multi-objective optimization is involved as a result of multiple objective functions considered in this paper. Due to this, a weighted sum approach is employed to give a better control on the final output based on our preferment. The approach is expressed as follows:

$$\sum_{i=1}^{M} w_i f_i(x) \quad ; w_1 f_1(x) + w_2 f_2(x) + \cdots + \omega_n f_n(x) \tag{8}$$

For the general purpose, we need to normalize all three objective functions in equation (8) to provide constant proportion for each objective function. This can be accomplished by distributing the fitness value with the maximum value for every objective function being measured. After applying weighted sum approach, the normalized fitness functions represented as below:

$$F(X) = w_1 f_1'(x) + w_2 f_2'(x) + w_3 f_3{}'(x) \tag{9}$$

Optimization Algorithm

In general, MAs share the same approach to their application for a given problem. First and foremost, the problem needs some representation in accordance with each method. Then, meta-heuristic search algorithms are used iteratively to reach a solution that is near-optimal. Due to basis on literature review that govern mixed-model assembly line balancing problem, GA, PSO, ACO, and SA are the most studied algorithm by previous researchers. This can be ranked by GA with the most used algorithm in solving MMALBP, then followed by SA, ACO, and PSO in second, third and fourth respectively. This is among the possible reason why these four MAs was chosen for comparison [14]. The following subsections presented a brief description governing this four MAs framework.

Ant colony optimization

ACO algorithms evolve not in their designated genetics only, but also in their social behavior. Look back in history of ACO, it led to Marco Dorigo [15] who first developed this algorithm. Taken from the metaphor on how the ants are able to search their source of food and nest by using the shortest route. The real framework on ACO algorithm is by using a pheromone trails, which scientifically deposited by ants when they navigate to find sources of food. This pheromone trails are used as some sort of communication medium between the ants.

As shown in **Error! Reference source not found.**, at the point when ants leave their homes looking for sources of food, they arbitrarily turn around an obstacle, and on the primary store of pheromone will be the same for the left and right directions. However, when the ants in the shorter direction discover the food, they carry it together and start returning back, following their pheromone trails, and still spare more pheromone. As indicated in this figure, an ant will most likely choose the shortest route when returning back to their home with food as this path will have the most deposited pheromone.

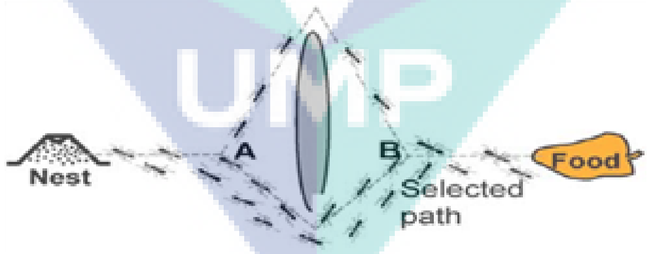


Figure 3 Ants searching for their food [6]

Particle swarm optimization

Based on the original literature, PSO was originally invented in the mid-1990s by Kennedy and Eberhart [16]. PSO is inspired by the behavior of flocks of bird in their journey to find the sources of food. Their social behavior helps them to adapt to the current environment as well as avoiding predators by using an approach called 'information sharing', hence, created an evolutionary advantage.

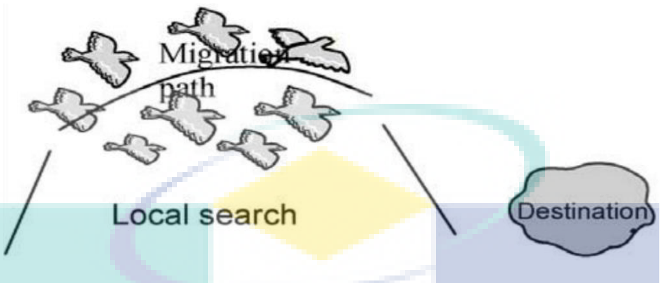


Figure 4 Flocks of bird looking for ther destination [6]

Genetic algorithms

GA recorded as the first evolutionary algorithm presented by John Holland in 1970s. Inspired from the principal of 'survival of the fittest', it is developed in a way which over a number of generation, the populations evolved by following the concept. Naturally, individual which possess a highest survival rate is likely to have a larger number of offspring. Hence, in each succeeding generation, the genes from the fittest individuals will increase in number. In this manner, species become more and more well-adapted to their current environment as they evolve [17].

Figure 5 Concept of survival of the fittest (i.e., tallest) [6]

Simulated annealing

SA algorithm is a meta-heuristic search technique which first invented by Kirkpatrick, Gelatt, and Vecchi in 1983. It serves a purpose for solving the NP-hard optimization problems, specifically to enhance the objective functions value. In fact, the 'annealing' term comes from the concept of annealing process used in metallurgical industry. Annealing is a process of slow cooling cast-off to metals in order to get a low energy-state crystallization and produce a better aligned finished metal product. The optimization procedure of SA searches for a near-optimum solution impersonating the slow cooling procedure in the physical annealing process [18].

Results and Discussions

In order to evaluate the performance of ACO, GA, PSO, and SA algorithm to an extend limit, a benchmark dataset in MMALBP must be tested. The test problems used in this paper is taken from the website http://www.assembly-line-balancing.de under the categories of mixed-model assembly line balancing problem. In addition, this test problem is widely used to test the algorithm in searching for quality solution to MMALBP.

The dataset contains small-size test problem (STP), medium-size test problem (MTP) and large-size test problem (LTP) ranging from 8 to 100 number of tasks. Specifically, small size problem contains 8 tasks to 20 tasks, medium size problem ranges from 25 to 50 tasks and large size problem from 60 to 100 number of tasks. All these four algorithms are developed based on its own features and targeted to optimize the selected objective functions.

Hence, all these four MAs is being tested by using the chosen test problem using MATLAB simulation. Environment of the computational experiment including: Intel(R) Core(TM) i7 2.40GHz, 8 GB memory, Windows 8.1. Considering that all four algorithms might be influenced by random characteristics, each optimization process is run for 20 times under the same parameter and experimental environment. Mean from the test result of each algorithm are listed. The result from this comparison of performance is presented as follows:

Table 1 MAs computational result (STP)

Small size (8-20 tasks)					
Problem(no.of task)	Algorithm	PSO	ACO	SA	GA
Bowman(8)	Min. Fitness	0.9443	**0.8166**	0.9443	**0.8166**
	Max.Fitness	0.9443	**0.8166**	0.9443	0.8193
	Mean Fitness	0.9443	**0.8166**	0.9443	0.8169
	Std.Deviation	0.0000	0.0000	0.0000	0.0007
	Mean Cputime	**82.9557**	88.0041	98.2792	89.3140
Mansoor(11)	Min. Fitness	0.7051	**0.5773**	0.7051	**0.5773**
	Max.Fitness	0.7051	**0.5773**	0.7051	0.5787
	Mean Fitness	0.7051	**0.5773**	0.7051	0.5775
	Std.Deviation	0.0000	0.0000	0.0000	0.0004
	Mean Cputime	**49.3870**	107.9176	161.7723	143.5887
Instance_small(20)	Min. Fitness	**0.7026**	0.9499	0.9443	0.8166
	Max.Fitness	**0.9077**	0.9513	0.9443	0.8179
	Mean Fitness	**0.8120**	0.9500	0.9443	0.8166
	Std.Deviation	0.0727	0.0004	**0.0000**	0.0003
	Mean Cputime	**52.3941**	131.1931	161.1315	140.1623

Table 2 MAs computational result (MTP)

Medium size (25-50 tasks)					
Problem(no.of task)	Algorithm	PSO	ACO	SA	GA
Buxey(29)	Min. Fitness	0.7138	**0.6395**	0.6538	0.8658
	Max.Fitness	0.9634	**0.7517**	0.8493	0.9268
	Mean Fitness	0.8401	**0.6969**	0.7906	0.8800
	Std.Deviation	0.0454	**0.0272**	0.0354	0.0275
	Mean Cputime	**101.4522**	101.7651	264.7180	243.4568
Gunther(35)	Min. Fitness	0.5888	0.7368	**0.5842**	0.8318
	Max.Fitness	0.9742	**0.8427**	0.8743	0.9671
	Mean Fitness	0.7973	0.7792	**0.6199**	0.9242
	Std.Deviation	0.1101	**0.0351**	0.0673	0.0599
	Mean Cputime	137.1709	**136.9874**	257.1917	153.3631
Instance_medium(50)	Min. Fitness	**0.5673**	0.6888	0.6499	0.6182
	Max.Fitness	**0.7318**	0.8304	0.7834	0.7723
	Mean Fitness	0.6359	0.7600	0.7162	**0.6106**
	Std.Deviation	0.0464	**0.0204**	0.0382	0.0697
	Mean Cputime	**276.8158**	337.7413	658.4571	405.0386

Table 3 MAs computational result (LTP)

Large size (65-100 tasks)					
Problem(no.of task)	Algorithm	PSO	ACO	SA	GA
Wee-mag(75)	Min. Fitness	0.6947	**0.6475**	0.6742	0.6505
	Max.Fitness	0.8411	0.7859	0.8247	**0.7595**
	Mean Fitness	0.7690	**0.7513**	0.7630	0.7777
	Std.Deviation	0.0397	**0.0329**	0.0393	0.0458
	Mean Cputime	1038.8310	**1024.2880**	1090.0520	1365.3380
Arc(83)	Min. Fitness	0.5274	**0.3807**	0.5279	0.7264
	Max.Fitness	0.6475	**0.4835**	0.6503	0.9647
	Mean Fitness	0.5903	**0.4412**	0.5832	0.7953
	Std.Deviation	0.0316	**0.0286**	0.0290	0.0614
	Mean Cputime	**1632.6530**	2631.1140	3044.7990	6313.9020
Instance_large(100)	Min. Fitness	0.5527	**0.4858**	0.4974	0.5772
	Max.Fitness	0.7298	**0.6743**	0.7358	0.8384
	Mean Fitness	0.6525	**0.6107**	0.6161	0.7012
	Std.Deviation	0.0517	**0.0510**	0.0710	0.0874
	Mean Cputime	**2148.1490**	2389.1640	2897.1870	3650.2650

Based on the result presented in **Error! Reference source not found.**, **Error! Reference source not found.**, and **Error! Reference source not found.**, the value which being bold are the best value in each parameter being measured. In the perspective of problem category included small, medium and large size test problem, each MAs resulted a different solution quality which will be discussed next.

Referring to STP in **Error! Reference source not found.**, the result are varies among ACO, PSO, SA and GA. The results being measured based on minimum fitness, maximum fitness, mean fitness, standard deviation and CPU time. For instance, in Bowman’s problem, ACO and GA gives the best value for minimum fitness meanwhile for maximum fitness and mean fitness ACO performed better compared to the other three MAs. On the other hand, PSO mean CPU time is the most superior in this problem. Likewise, the same result analysis are produced for Mansoor’s problem. However, the different result was found for the Instance_small’s problem with 20 number of tasks. PSO at its best performance in this problem when performed better for minimum fitness, maximum fitness, mean fitness and mean CPU time than the other three MAs. The only parameter its lacking is standard deviation which give an early view eventhough its performed better but the solution quality may vary for each single run.

Moving from STP, MTP analysis diplayed in **Error! Reference source not found.** gives us more diversified result. For example, better solution quality for Buxey’s problem pointed to ACO algorithm but mean cpu time come off second-best to PSO. Meanwhile for Gunther’s problem, SA algorithm outperformed its competitors by yielded better minimum fitness and mean fitness but lack to ACO in term maximum fitness, standard deviation, and mean CPU time. Next in Instance_medium's problem, PSO algorithm offered better minimum fitness, maximum fitness and mean CPU time whereas ACO present better standard deviation with GA produce the best mean fitness.

Result tabulation in **Error! Reference source not found.** represent LTP which contain the highest number of tasks in this study ranging from 65 to 100 tasks. Wee-Mag's problem shows that ACO algorithm achieved better result for each parameter except maximum fitness which come off second-best to GA. Based on Arc's problem with 83 number of tasks, ACO algorithm performed better in terms of minimum fitness, maximum fitness, mean fitness, and standard deviation than the other three MAs. The only shortcomings is in term of its mean CPU time which trailing behind PSO algorithm. Lastly, all four MAs are tested using Instance_large's problem contained 100 number of tasks. Once again ACO algorithm give a result with a better solution quality and only lack in term of mean CPU time to PSO. In addition, LTP gives a better view on which MAs finish the optimization with a constant solution quality over a multiple run. The best standard deviation value in LTP clearly dominated by ACO algorithm. Hence presented an early view on term of its solution reliability compared to PSO, GA, and SA.

There is a pattern can be identified in this experiment which is PSO and ACO performed better when dealing with STP and MTP covering a range from 8 till 50 number of tasks. However surprisingly, when being tested with LTP, PSO algorithm's performance is decreased. On the other hand, ACO algorithm proved its reliabilty when solving LTP compared to the others three MAs. The phenomenon happened to PSO algorithm in this experiment however can be related to the mechanisms of PSO itself. Since LTP is used, it brought a larger solution space with it that need to be explored. The designated inertia, ω in PSO framework takes place and affect the solution quality over generation. Generally, ω is equal to 1, then at the later period of the several generations, there is a lack of the searching ability of the particle for a better solution quality which produced poorer result for PSO [19]. In order to calculate the percentage difference of performances by this four MAs in every category, an average value for best fitness solution is tabulated in Table 4.

Table 4 Mean of solution quality by each MAs for every categories

	Problem Category	Algorithm			
		ACO	PSO	GA	SA
Avg. Solution quality	STP	0.7813	0.8204	**0.7703**	0.8645
	MTP	**0.7453**	0.7577	0.8049	0.7789
	LTP	**0.6010**	0.6705	0.7581	0.6541
Avg. CPU time (min)	STP	327.114	**184.736**	373.065	421.183
	MTP	776.493	**705.438**	801.858	1662.367
	LTP	6044.56	**4819.633**	7329.505	9032.038

*An Intel Core i7-3630QM@2.4Ghz was used

As mentioned earlier, each optimization for each MAs is repeated 20 times and each single run returned the best fitness. Hence, to plot the convergence graph, mean value of solution output for each algorithm is used. Figure 6 Convergence plot for MAs in STPFigure 6, Figure 7, and Figure 8 present the mean convergence of four MAs for each categories.

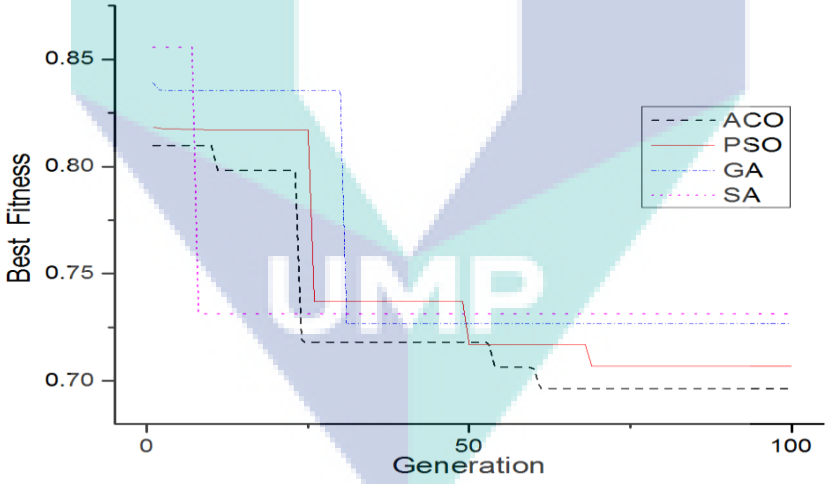


Figure 6 Convergence plot for MAs in STP

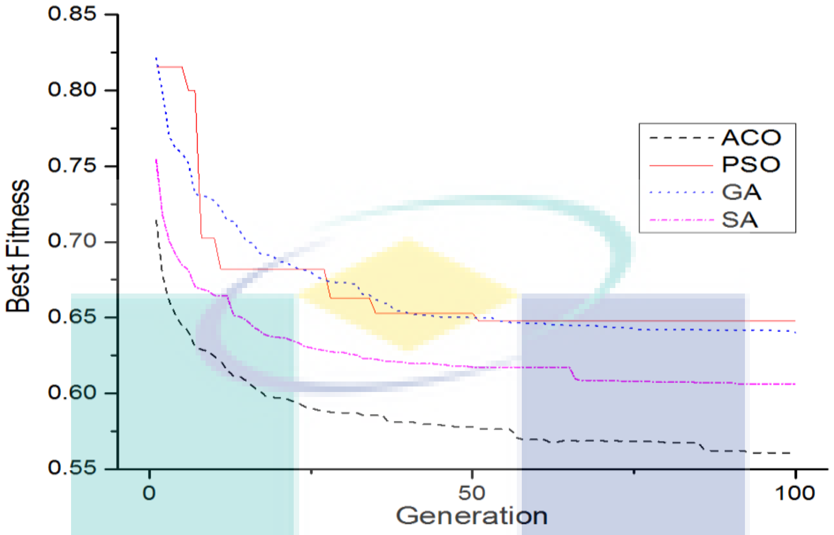


Figure 7 Convergence plot for MAs in MTP

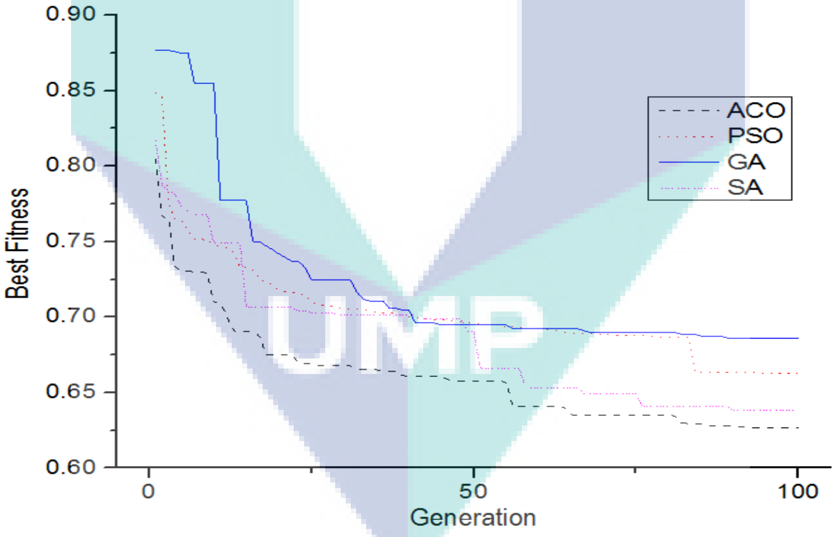


Figure 8 Convergence plot for MAs in LTP

Based on the graph in Figure 6, all the algorithms show rapid convergences, even though the ACO presents better performance. The rapid convergence in this class of problem is due to the small search space in small size problem. In contrast, with the increasing number of tasks from small size to medium and large size problem, the search space is become larger. Therefore, the convergence was occured more frequent since the choices of solutions were much larger.

Conclusion

The performance represented by fitness function and processing time of four meta-heuristic algorithms (ACO, PSO, GA, and SA) in MMALBP with resources constraint has been presented. The comparison between these MAs were made and selected objective functions has been evaluated which are to minimize cycle time, minimize product rate variation and minimize resources used on the assembly line. One of the significant findings to emerge from this study suggested that ACO algorithm performed better in term of finding the best fitness functions when dealing with a large number of tasks. Averagely, it has produces better solution quality than PSO by 5.82%, GA by 9.80%, and SA by 7.66%. However, PSO more superior in term of processing time than ACO by 29.25%, GA by 40.54%, and SA by 73.23%.

The present study however, has some limitation included the parameter settings for the algorithms. In this study, the parameter tuning is not considered. A proper parameter settings for algorithm could possibly return a better solution quality. Further investigation also can be implement by consider actual assembly line such as manufacturing industry. Based on the listed limitation, a further experimentation to measure the performance of these four MAs is strongly recommended. Besides that, it would be interesting to assess the effects of increasing the number of test problem to the solution quality produced as well as the results when applying to the actual problems in an actual assembly line.

Acknowledgement

This research has been supported by research fund from Ministry of Higher Education under RDU140103. In addition, the work was originally accepted for and presented at MERD'17.

Reference

[1] M. Rabbani, M. Moghaddam, and N. Manavizadeh, "Balancing of mixed-Model two-Sided assembly lines with multiple u-Shaped layout," *Int. J. Adv. Manuf. Technol.*, vol. 59, no. 9–12, pp. 1191–1210, 2012.

[2] Z. Yuguang, A. Bo, and Z. Yong, "A PSO algorithm for multi-objective hull assembly line balancing using the stratified optimization strategy," *Comput. Ind. Eng.*, 2016.

[3] I. Kucukkoc and D. Z. Zhang, "Mathematical model and agent based solution approach for the simultaneous balancing and sequencing of mixed-model parallel two-sided assembly lines," *Int. J. Prod. Econ.*, vol. 158, pp. 314–333, 2014.

[4] O. Battaïa, X. Delorme, A. Dolgui, J. Hagemann, A. Horlemann, S. Kovalev, and S. Malyutin, "Workforce minimization for a mixed-model assembly line in the automotive industry," *Int. J. Prod. Econ.*, vol. 170, pp. 489–500, 2015.

[5] X. Zhao, C.-Y. Hsu, P.-C. Chang, and L. Li, "A genetic algorithm for the multi-objective optimization of mixed-model assembly line based on the mental workload," *Eng. Appl. Artif. Intell.*, vol. 47, pp. 140–146, 2015.

[6] E. Elbeltagi, T. Hegazy, and D. Grierson, "Comparison among five evolutionary-based optimization algorithms," vol. 19, pp. 43–53, 2005.

[7] Sener and G. Mirac Bayhan, "A hybrid genetic algorithm for mixed model assembly line balancing problem with parallel workstations and zoning constraints," *Eng. Appl. Artif. Intell.*, vol. 24, no. 3, pp. 449–457, 2011.

[8] N. Manavizadeh, N. Hosseini, M. Rabbani, and F. Jolai, "A Simulated Annealing algorithm for a mixed model assembly U-line balancing type-I problem considering human efficiency and Just-In-Time approach," *Comput. Ind. Eng.*, vol. 64, no. 2, pp. 669–685, 2013.

[9] M. M. Razali, M. Fadzil, F. Ab, and M. Razif, "Mathematical Modelling of Mixed-Model Assembly Line Balancing Problem with Resources Constraints," vol. 12002.

[10] B. Yagmahan, "Mixed-model assembly line balancing using a multi-objective ant colony optimization approach," *Expert Syst. Appl.*, vol. 38, no. 10, pp. 12453–12461, 2011.

[11] N. Manavizadeh, L. Tavakoli, M. Rabbani, and F. Jolai, "A multi-objective mixed-model assembly line sequencing problem in order to minimize total costs in a Make-To-Order environment, considering order priority," *J. Manuf. Syst.*, vol. 32, no. 1, pp. 124–137, 2013.

[12] Y. Kara, U. Ozcan, and A. Peker, "Balancing and sequencing mixed-model just-in-time U-lines with multiple objectives," *Appl. Math. Comput.*, vol. 184, no. 2, pp. 566–588, 2007.

[13] A. a. Mamun, A. a. Khaled, S. M. Ali, and M. M. Chowdhury, "A heuristic approach for balancing mixed-model assembly line of type I using genetic algorithm," *Int. J. Prod. Res.*, vol. 50, no. 18, pp. 5106–5116, 2012.

[14] R. Hassan and B. Cohanim, "A comparison of particle swarm optimization and the genetic algorithm," *1st AIAA Multidiscip. Des. Optim. Spec. Conf.*, pp. 1–13, 2005.

[15] M. Dorigo and G. DiCaro, “The Ant Colony Optimization meta-heuristic,” pp. 11–32, 1999.
[16] J. Kennedy and R. Eberhart, “Particle swarm optimization,” *Neural Networks, 1995. Proceedings., IEEE Int. Conf.*, vol. 4, pp. 1942–1948 vol.4, 1995.
[17] J. H. Holland, *Adaptation in Natural and Artificial Systems*, no. 1. MIT CogNet, 1992.
[18] A. Hamzadayi and G. Yildiz, “A simulated annealing algorithm based approach for balancing and sequencing of mixed-model U-lines,” *Comput. Ind. Eng.*, vol. 66, no. 4, pp. 1070–1084, Dec. 2013.
[19] Q. Bai, “Analysis of Particle Swarm Optimization Algorithm,” *Comput. Inf. Sci.*, vol. 3, no. 1, pp. 180–184, 2010.

Optimization of Two-sided Assembly Line Balancing with Resource Constraints using Modified Particle Swarm Optimization

Muhammad Razif Abdullah Make[1], Nur Hairunnisa Kamaruddin[1], Muhamad Magffierah Razali[1] and Mohd Fadzil Faisae Rashid[1],*

[1]Manufacturing Focus Group, Faculty of Mechanical Engineering, Universiti Malaysia Pahang, 26600, Pekan, Pahang, Malaysia.

*Corresponding Author's Email: ffaisae@ump.edu.my

Abstract

Two-sided Assembly Line Balancing (2S-ALB) is important in the assembly plants that produce a large-sized high-volume product, such in the automotive production. Many of the existing publication on 2S-ALB however, ignored the assembly resources such as worker skills, tools and machines that required for the assembly in the problem modeling. This research model and optimize the 2S-ALB with resource constraints. In the end, besides having good workload balance, the number of resources also can be optimized. For optimization purpose, Particle Swarm Optimization is modified to reduce the dependencies on a single best solution. This is conducted by replacing the best solution with top three solutions in the reproduction process. Computational experiment result using 12 benchmark test problems indicated that the 2S-ALB with resource constraints model able to reduce the number of resources in an assembly line. Furthermore, the proposed modified Particle Swarm Optimization (MPSO) capable to search for minimum solutions in 11 out of 12 test problems. The good performance of MPSO is because of ability to maintain the particle diversity over the iteration. In future, Pareto optimality concept is proposed to deal with multi-objective optimization problem.

Keywords: Assembly line balancing, Two-sided, Particle Swarm Optimization

1. Introduction

Assembly line is a system that considered the arrangement of workstation, workers, tools or machine that successively outline the operation for being completed. It has been widely used in many manufacturing industries to cope with the increase of demands in manufacturing. The assembly line is set up for the most optimum design to meet the production demand. The assembly line system was introduced around 1900 by Henry Ford for his automobile plants [1]. Since then, various evolutions and progress has been reported towards the assembly line. Commence from that idea, the balancing approach has been developed for the assembly line, known as Assembly Line Balancing (ALB). Balancing an assembly line can be difficult for most of the industries. It not only refers to assigning each of the tasks into the workstation but also concern towards to enhance the production rate with desired performance level [2]. Nowadays, the ALB has become more important to cope with global competitiveness in the industry. It classically started in 1955, when Salveson firstly describe the typical ALB problem focusing on efficient and fast solution approach for solving the line balancing problem [3]. The great progress developed from time to time has extended the classification of the ALB problem.

Later, various version of ALB problems has been formulated to suit with different assembly line problems. One of the ALB branches is the assembly line that assembled large-sized and high-volume product like an automotive assembly line. The assembly process is conducted on both left and right sides of the product. This problem is known as two-sided assembly line balancing (2S-ALB). This problem was first exposed by Bartholdi in 1993 [4]. This early work on 2S-ALB has inspired the other researcher to study and extend this work to the next level. The 2S-ALB was built from a single line production system which identically paired parallel to the first sided of the assembly line. **Figure 1** illustrates the 2S-ALB station features along conveyor belt. Different from one sided line, the assembly process in 2S-ALB could be conducted either from the left or right side, depends on various constraints. The 2S-ALB system able to shorten and saves some space of the assembly lines, besides reduce the material handling of tools and fixture.

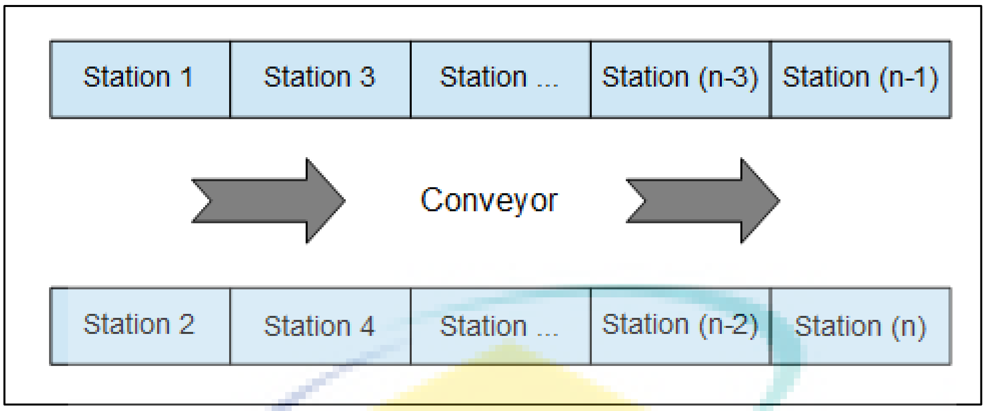


Figure 1. Two-sided assembly line

Nowadays, the 2S-ALB problem has widely grown and adopting different ALB version as the variation of the 2S-ALB problem. The 2S-ALB variation started with the general of 2S-ALB as illustrated in **Figure 1**. The general 2S-ALB consists of two workstations facing each other along the assembly line. This version of the problem has its advantages such as shorten the assembly line, save some spaces, reduce throughput time and material handling, besides the cost of tools and fixtures. This general 2S-ALB is well addressed in several research studies [3], [5]–[8].

Besides studying on the general 2S-ALB, researchers also combined the 2S-ALB with the mixed-model assembly line balancing (MALB). The MALB its particularly considered to level the workload in every workstation on the line, besides leveling the part usage. It literally to achieve a balanced workload at specific processing times for each assembly task, while attempting to minimize the variation used by the different parts over time. This combination of 2S-ALB with MALB has broadly introduced implementing towards different optimization and line balancing solution approach [9]–[11]. Another combination with the 2S-ALB was parallel assembly line balancing (P-ALB). The P-ALB is the combination of two or more lines placed parallel to each other became an idea of sharing tools and fixtures to complete the entire job. The two-sided parallel assembly line balancing which is the combination of 2S-ALB and P-ALB is to shorten the assembly line while steadily runs during breakdown [12]–[15]. This combined problem was discussed by Ozcan, Gokcen, and Toklu (2010) [16] providing much more benefits: (i) It can help to produce similar products or different models of the same production of the adjacent lines. (ii) It can reduce the idle time and increase the efficiency of the assembly lines. (iii) It is able to make production with a different cycle time for each of the lines. (iv) It can improve visibility and communication skills between operators. (v) It is also able to reduce operator requirements.

Many studies have been conducted to work out with the best optimum seeking approach implementing either heuristic or meta-heuristic method for 2S-ALB. In the early study, Kim et al. in 2000 used Genetic Algorithm (GA) as the optimization algorithm [17]. Then in 2001, it has been continued by Lee, Kim et al. employing group assignment procedure [18]. The GA approach was also implemented by [15], [19], [20] to optimize 2S-ALB. Meanwhile, Baykasoglu and Dereli adopt Ant Colony Optimization (ACO) to optimize the 2S-ALB [21]. They have successfully applied the ACO algorithm for a domestic product which influences the other researcher to deal with other sector apart from the large-sized automotive products. Apart from that, many other researchers also implemented the ACO because of the good performance for the combinatorial problem [14], [22], [23]. From the earlier review, GA and ACO algorithm are successfully dominate the other optimization methods in term of performance and also frequencies that make these algorithms more popular [12]. Besides that, different algorithms were also implemented through several reported research. For instance, Hu et al. (2008) were reported to implement the enumerative algorithm combined with the Hoffmann heuristic method [24].

In the meantime, Particle Swarm Optimization (PSO) algorithm was also frequently implemented for 2S-ALB. The PSO assisted with Taguchi has been implemented for 2S-ALB with multi-skilled worker assignment [25]. Researchers also implement ACO algorithm to optimize stochastic 2S-ALB, instead of deterministic time in the majority of 2S-ALB works [26]. While in 2012, Chutima and Chimklai proposed a Particle Swarm Optimization with Negative Knowledge (PSONK) for the optimization of complex combination with the 2S-ALB problem [10]. [27] later also implement the PSONK, but proposing a combined selection mechanism for the assembly task. Besides that, different approaches have been proposed to improve the PSO performances [28]–[30]. Although the advantage of PSO algorithm has been well reported, its application and improvement are still needed. Generally, PSO is known as the fast optimizer and a robust algorithm which provide a high quality of the solution. However, the highly focused towards a single best solution in PSO can lead to a premature convergence or local optima. This phenomenon is described where the convergence is stopped earlier and considers to express the solution as the best. This problem occurred when the algorithm tried to figure out the path of searching direction, while still following the best earlier solution. The limitation in providing the best solution might occur due to the fewer parameter setting as the PSO algorithm only require a simple specification as a setting before generating a solution.

Despite there were many studies on 2S-ALB have been conducted, the majority of these works assumed the assembly workstation has similar capability to conduct the assembly process. In a real situation, there are various constraints that need to be considered during the assembly line design. For example, the workforce and machine that have different skills and ability in completing the assigned task. The proper utilization of resource depending on their skills and precedence has integrated the assembly line for to be fully optimized. Besides, with the appropriate use of the machine, it's also able to solve the inadequate space problem for the assembly line in allocating the required machines on the workstation [31].

In order to overcome the limitation, this paper will consider the resources that required to conduct a specific assembly task. By considering the assembly resource constraints, the number of resources could be optimized. For optimization purpose, the PSO will be modified to reduce the dependence of algorithm on a single best solution. The proposed modification is expected to improve the algorithm exploration ability. Section 2 of this paper presents the 2S-ALB with resource constraints problem. Section 3 presents the proposed Modified Particle Swarm Optimization (MPSO) algorithm. The computational experiment set up and results are discussed in Section 4. Finally, Section 5 summarize and conclude the research work.

2. 2S-ALB with Resource Constraints Problem

The 2S-ALB is modified structure that essentially born from one-sided ALB problem. The main goal of this problem is to enhance the production rate and increase the line efficiency. Flexibility to produce a high volume of large-sized product in two-sided assembly line configuration practically provides many beneficial advantages such able to shorten the line length, save spaces on the lines, increase the line efficiency by reducing the number of workstation, and able to reduce the material handling cost of tools and fixture. Normally in a two-sided assembly line, there will represent by a pair of lines placed opposite to each other. **Figure 1** has illustrated the two-sided assembly line possessed left and right side of the lines in which the workstation are clamped together between the moving conveyor.

The comprehensive study in making the idea of balance 2S-ALB problem has been presented in an earlier study [2]. Commence from a particular task relation called ‘precedence relation graph’ that built with circle and arrows. The example of precedence relation graph with nine tasks is depicted in **Figure 2**. Each circle represents as the assigned task, while the arrows linked represent each relation between the task. The associated data of each processing time and operational direction also specified on top of each circle (assign task). Three type of operational direction will be considered: left (L), right (R) and either (E). For left and right side the execution is outright and should be actualized for the following position. Meanwhile, for either side direction, the task could be executed on any side of the workstation either on the left side or the right side.

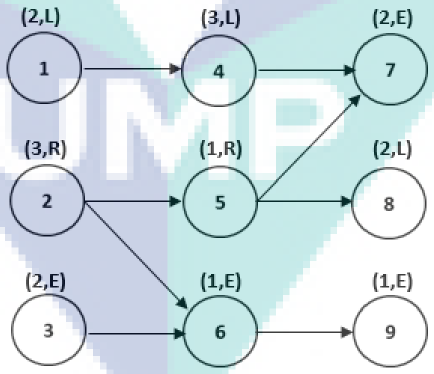


Figure 2. Precedence relation graph

Then, an assembly data is presented in precedence matrix as shown in **Table 1**. This matrix consists of ones and zeros values that represent the assembly relation information of the precedence graph. In **Table 1**, the relation of each task is transformed from the precedence relation graph, adopting '*i*' as the present task and '*j*' as the next assigned task. The value of one in the precedence matrix indicates the predecessor link of '*i*' task to the next task '*j*'. It means that there is a precedence relationship to be examined. Meantime the zeros value implies no precedence relation between task *i* and *j*.

Table 1. Precedence matrix

i/j	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	0	0	0	1	0	0	0	0	0
2	0	0	0	0	1	1	0	0	0
3	0	0	0	0	0	1	0	0	0
4	0	0	0	0	0	0	1	0	0
5	0	0	0	0	0	0	1	1	0
6	0	0	0	0	0	0	0	0	1
7	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0

Table 2 Data matrix

Task	**Time**	**Side**	**Resources**		
1	2	1	1	2	0
2	3	3	3	0	0
3	2	2	2	3	0
4	3	1	1	0	0
5	1	3	3	0	0
6	1	2	2	3	0
7	2	2	1	2	3
8	2	1	2	0	0
9	1	2	1	3	0

Besides the precedence matrix, a data matrix is also required to store the assembly information for the 2S-ALB with resource constraints. The data matrix (**Table 2**) express the assembly information such as processing time, assembly side and resources detail. For side column, three different operational direction value indicates different sides. In this column, value ‘1’ for left side operation, ‘2’ for either side operation while ‘3’ for the right side operation. The resources detail also coded in numbers to express different resources. It is important to note that the number of resources for one assembly task is not limited to three as shown in **Table 2**. In the case where the number of resources is larger, the matrix can be expanded to fit all the data.

2.1 Problem Assumptions and Notations

The general assumptions of the problem are as follows:

- Task times and resource used (machine, tools, worker) are known and deterministic.
- Tasks have preferences regarding the operational direction (side), i.e., left side, either side or right side.
- Every task can be operated only after all its immediate predecessors are completed.
- The maximum operational cycle time is fixed and could not be exceeded.
- Every task cannot be split between workstations and must be assigned to exactly one workstation.
- The tasks with positive zoning must be operated in the same workstation.
- The tasks with negative zoning could not be assigned to the same workstation.
- Parallel tasks and parallel stations are not allowed
- The skills level of each worker is ignored to provide the similar working pace of assembly task.
- The working travel times are ignored and no inventory (work in progress) is allowed.
- Any breakdowns of machine and tools are not considered and the assembly process is constantly performed.

The notations used in this mathematical formulation are summarized as follows.

J	: number of mated-workstation $j = 1, 2, \dots, J$
I	: number of one-sided workstation $i = 1,2, \dots, I$

F	: 1, if there is any space availabile on the operating time, otherwise, 0
N	: number of resource utilization $n = 1,2, \dots, N$
X_{ms}	: 1, if mated-workstation j is utilized for both side of the line, otherwise, 0
Y_s	: 1, if mated-workstation j is utilized for only one side of the line, otherwise, 0
m_t	: maximum processing time $t = 1,2, \dots, T$
r_t	: operational time of the task on the workstation j
p_v	: maximum gap value in space availability
q_v	: minimum gap value in space availability
R_s	: 1, if resource is utilized in workstation j, otherwise, 0

2.2 Mathematical Formulation and Constraints

The mathematical model for 2S-ALB with resource constraints are presented below. In this problem, four optimization objectives are considered. The first optimization objective as in equation (1) is to minimize the mated workstation, f_1. The second optimization objective in equation (2) is to minimize the number of the workstation, f_2. A mated workstation consists of a pair of left and right workstation on the assembly line. Meanwhile, the number of workstation calculate the total individual workstation. The third optimization objective is to minimize idle time, f_3 as presented in equation (3). Finally, the optimization objective to minimize the number of resources, f_4 is presented in equation (4). By using the number of resources as one of optimization objective, the number of the resources can be minimized. This can be achieved by assigning the assembly task that used a similar resource in one workstation.

$$f_1 = \sum_{j=1}^{J} X_{ms} \tag{1}$$

$$f_2 = \sum_{j=1}^{J} 2JX_{ms} + \sum_{i=1}^{I} Y_s \tag{2}$$

$$f_3 = \sum_{t=1}^{T} (m_t - r_t) + \sum_{t=1}^{T} F(p_v - q_v) \tag{3}$$

$$f_4 = \sum_{n=1}^{N} R_s \tag{4}$$

$$\sum_{K} k(x_{ik1} + x_{ik2}) - \sum_{K} k\big(x_{jk1} + x_{jk2}\big) = 0 \qquad (i,j)\ \epsilon\ ZP_{ij} \tag{5}$$

$$\sum_{K} k(x_{ik1} + x_{ik2}) - \sum_{K} k\big(x_{jk1} + x_{jk2}\big) \neq 0 \qquad (i,j)\ \epsilon\ ZN_{ij} \tag{6}$$

$$\sum_{i=1}^{n} t_i x_{ijk} + s_{jk} \leq CT \tag{7}$$

$$s_{jk} = \sum_{u=1}^{U} x_{ujk}\big(t_{u+1}^{s} - t_{u}^{f}\big) + \big(CT - t_{u}^{f}\big) \quad u\ \epsilon\ Q_{jk} \tag{8}$$

$$\sum_{k\epsilon\{1,3,5,\ldots,m-1} x_{jk} = 1 \ \ \forall j\ \epsilon\ L \tag{9}$$

$$\sum_{k\epsilon\{2,4,6,\ldots,m\}} x_{jk} = 1 \ \ \forall j\ \epsilon\ R \tag{10}$$

$$\sum_{k=1}^{m} x_{jk} = 1 \ \ \forall j\ \epsilon\ E \tag{11}$$

Besides the optimization objectives in equation (1) to (4), several constraints also being considered to ensure the feasibility of generated solution. Constraint (5) enable different task for to be assigned to the same workstation. Meanwhile, constraint (6) limits the assigned task on the same workstation as different prescribed equipment. Constraint (7) and (8) are related to controls and ensure the maximum operational cycle time for not be exceeded. Constraint (9), (10) and (11) engaged to assigned each task to only one workstation which is either left or right.

In this work, weighted sum approach is used to deal with the multi-objective problem. Therefore, the optimization objectives that considered in this work need to be normalized because they have different ranges. For this purpose, the f_i is normalized into [0, 1] range as follow:

$$\widehat{f_i} = \frac{f_i - f_{i_{min}}}{f_{i_{max}} - f_{i_{min}}} \tag{12}$$

The minimum and maximum optimization objectives are defined as follow:

$$f_{1_{min}} = 0 \tag{13}$$

$$f_{1_{max}} = f_{2_{min}} \tag{14}$$

$$f_{2_{min}} = \frac{\sum_{i=1}^{n} t_i}{ct_{max}} \tag{15}$$

$$f_{2_{max}} = \frac{\sum_{i=1}^{n} t_i}{\max\limits_{i=1:n}(t_i)} \tag{16}$$

$$f_{3_{min}} = 0 \tag{17}$$

$$f_{3_{max}} = f_{2_{max}} . ct_{max} - \sum_{i=1}^{n} t_i \tag{18}$$

$$f_{4_{min}} = r_{type} - 1 \tag{19}$$

$$f_{4_{max}} = \sum r \tag{20}$$

The fitness function for this problem is presented as follow. The w_1, w_2, w_3 and w_4 is set at 0.25.

$$f = w_1\hat{f}_1 + w_2\hat{f}_2 + w_3\hat{f}_3 + w_4\hat{f}_4 \tag{21}$$

3. Modified Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a meta-heuristic searching method that inspired from the swarming behavior of flocking birds. This mechanism particularly respect to migrating birds population and its flying directions. For every single migrating bird is considered a particle which usually adjusts their searching or flying direction according to the previous flying experience. Each particle represents as the potential solution with a certain position (current solution), velocity (magnitude and direction towards the optimal solution) and fitness value (performance measure of the specific problem). While compared to another evolutionary approach such ACO and GA method, PSO is respectively known has a faster convergence towards the optimal solution [32].

The PSO algorithm begins with initialization procedure in which each particle represents the population in a *D*-dimensional vector as the constructed possible solution, $X_i = (x_{i1}, x_{i2}, \dots, x_{iD})$ and velocity, $V_i = (v_{i1}, v_{i2}, \dots, v_{iD})$. Then, each solution is evaluated according to the objective function. Since the PSO is coded using real number, a topological sort procedure is applied to match with combinatorial problem in 2S-ALB. For the example in **Figure 2**, let the X_1 = (4.81, 7.90, 2.12, 6.91, 6.63, 4.09, 0.27, 3.54, 3.95). The topological sort begin with identify the candidate task without precedence. In **Figure 2**, task 1, 2 and 3 are the candidate tasks. In this situation, the x_{11}, x_{12} and x_{13} is compared to determine the selected task. Since x_{12} is the highest, task 2 is selected and stored in feasible solution, F_1 = [2]. The selected task is then being removed from the precedence graph. This approach is repeated until all the task from the graph is selected. For this example, the decoded feasible solution is F_1 = [2 5 1 4 8 3 6 9 7].

Next, the particle best solution (Pbest) and global best (Gbest) are updated. Pbest refers to the current best solution for a particular particle, while the Gbest is the overall best solution. The Pbest and Gbest solutions are used to update the velocity and position of the solution. The following formula used to update velocity (22) and position (23):

$$V_i^{t+1} = wV_i^t + c_1r_1\left(Pbest_i^t - X_i^t\right) + c_2r_2(Gbest^t - X_i^t) \tag{22}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{23}$$

In equation (22), t denotes the iteration number, while w is the inertia weight for regulating the previous effect of historical velocities. On the other hand, c_1 and c_2, there are the acceleration coefficients, while r_1 and r_2 are the random number between [0, 1]. The Pbest, Gbest and particle position are updated until the specific iteration number is reached.

Previously, a lot of research papers proposed different approaches to reduce premature convergence in PSO. Premature convergence in soft computing occurred because lack of the diversity in the solution during the iteration process. In PSO, this phenomenon is directly related with velocity and position updating procedure. The solution position is influenced by the Pbest and Gbest with some randomness by r_1 and r_2. The Pbest however only influence a specific particle, compared with Gbest that affect all of the particles to move towards it. In the case where Gbest is not been updated (no better solution found) in a few consecutive iterations, there is a possibility for the majority of particles to reach the Gbest. This situation will reduce the solution diversity.

To overcome this problem, this work proposed to consider top three best solutions instead of the only single solution in Gbest. For this purpose, the single solution in Gbest is replaced with the average of three best solutions.

$$Gbest^t = (g_1^t + g_2^t + g_3^t)/3 \tag{24}$$

In equation (24) g_1^t, g_2^t and g_3^t refer to the solution particle in the first, second and third ranks respectively for the t^{th} iteration. In the modified PSO, the Gbest is replaced with the new Gbest in equation (24). The reason to consider three top solutions for Gbest is to improve the solution diversity. In the proposed mechanism, the particle position will follow the average position from three best solutions. Furthermore, the possibility for all three solutions not been updated is small compared with single Gbest solution in the original PSO. This mechanism makes the search direction become more diverse, and the chance to trap in local optima can be reduced.

To prove this concept, a simple test using Rastrigin function is conducted. For this function, the optimum point is (0, 0). In this test, only six particles are used. The first particle is set as (0, 0) while the remaining five particles are randomly generated using the same pseudorandom for both PSO and MPSO. The purpose to set the first particle as the optimum point is to observe the particle movement over the iteration. For this purpose, the iteration is only set to 10. The particle position for the first, fifth and tenth iteration are captured. All other parameters for PSO and MPSO are the same.

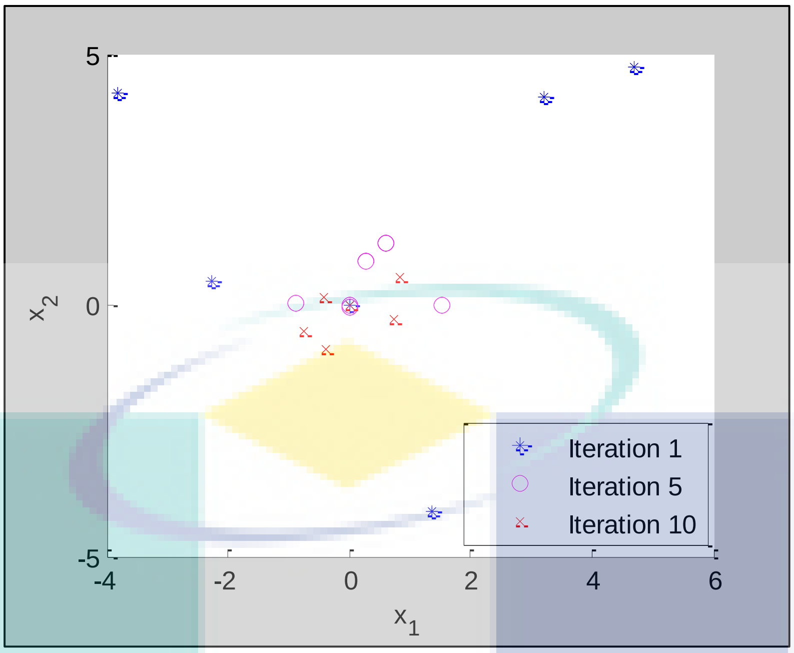


Figure 3(a). Particle movement for PSO

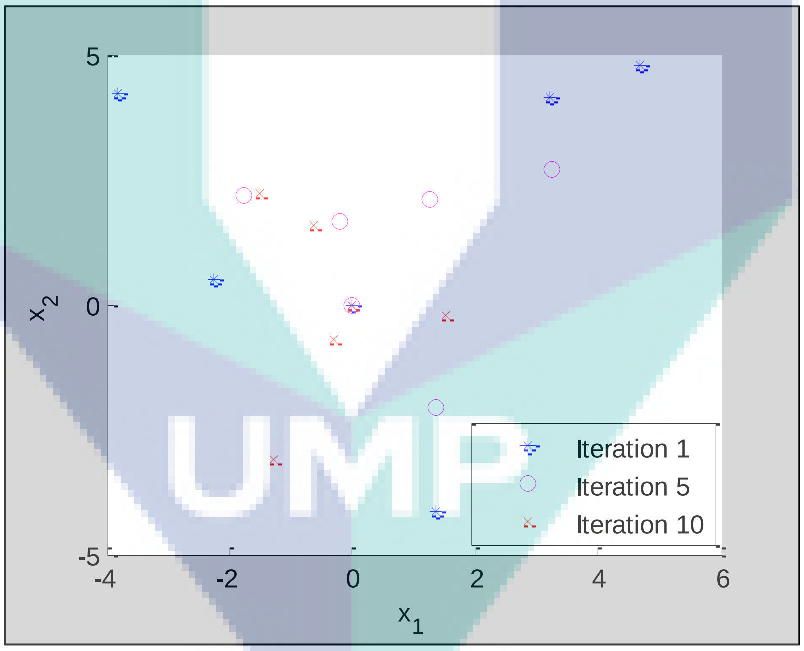


Figure 3(b). Particle movement for MPSO

Figure 3(a) and **3(b)** present the particle movement for PSO and MPSO. In **Figure 3(a)**, all particles move directly towards the Gbest (i.e. point (0, 0)) during the fifth iteration. During iteration 10, the particles only search the solution around the Gbest within a limited range. Meanwhile in MPSO, the particles capable to maintain the diversity in fifth and tenth iteration (**Figure 3(b)**).

Although the searching range over the iteration becomes smaller, the particles in MPSO not directly move towards the best solution. Therefore, it is expected that the MPSO will have better exploration ability. The procedure of MPSO is presented as follow:

Procedure of Modified PSO

```
Initialize MPSO parameters: Population size ($npop$), coefficients ($w$, $c_1$, $c_2$), iteration counter
    ($iter = 0$) and maximum iteration ($iter_{max}$)
Initialize random velocity, $V_i$ and position, $X_i$ for $i = 1,2,\ldots, npop$
While $iter \leq iter_{max}$
    $iter = iter + 1$
    Decode the $X_i$ into feasible assembly sequence, $F_i$
    Evaluate the fitness function for $i$th solution, $f_i$
        Update personal solution, $Pbest_i$
        Update top three global solutions, $g_1$, $g_2$ and $g_3$
        Update $Gbest = (g_1 + g_2 + g_3)/3$
    Update velocity
        $V_i^{t+1} = wV_i^t + c_1r_1(Pbest_i^t - X_i^t) + c_2r_2(Gbest^t - X_i^t)$
    Update position
        $X_i^{t+1} = X_i^t + V_i^{t+1}$
End
```

4. Results of Computational Experiment

A computational experiment is conducted to measure the performance of modified PSO (MPSO) to optimize 2S-ALB with resource constraints. For this purpose, 12 benchmark test problems are selected according to small, medium and large size. The test problems are adopted from different sources [3], [4], [7], [17], [18], [33], [34]. Based on the range of problem size used in literature, the small size problem is the assembly problem with less than 20 tasks. Meanwhile, the large size problem is the problem with more than 80 tasks. The assembly problem in between 20 to 80 tasks is considered as medium size. The detail of the test problems is presented in **Table 3**. Due to the lack of large size test problems, problem T83 and T111 are adopted from simple assembly line balancing problem and the assembly direction (i.e. left, right or either) are randomly generated. These benchmark problems however did not consider the resources required to conduct assembly task. Therefore, the assembly resources are also randomly generated for each of the assembly tasks.

Table 3. Test problem category and sources

Size	Problem	Number of task	Data source
Small	T4	4	[7]
	T9	9	[17]
	T12	12	[17]
	T16	16	[18]
Medium	T24	24	[17]
	T47	47	[33]
	T65	65	[18]
	T70	70	[3]
Large	T83	83	[34]
	T111	111	[34]
	T148	148	[4]
	T205	205	[18]

The performance of MPSO is then compared towards Genetic Algorithm (GA), Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO). These algorithms are chosen because of their popularity in optimizing 2S-ALB problem. According to the earlier survey on the ALB problem, 70% of the problem was optimized using GA, ACO and PSO algorithms [35]. The recent survey on 2S-ALB also reveals that the GA and ACO were the popular algorithms to optimize 2S-ALB according to the frequencies [12]. For the computational purpose, the population size for all algorithms is 30 and the maximum iteration is 500. The optimization run is repeated for 20 times with different pseudorandom for each of the cases.

The optimization results for the 2S-ALB with precedence constraints were presented in **Table 4** until **Table 6** based on the problem size. For the result of small size in **Table 4**, all algorithms able to generate the same fitness and objective function value for T4 and T9 problems. On the other hand, for T12 problem, the MPSO shows the best fitness compared with other algorithms. For this problem, the MPSO able to search for a solution with a smaller number of resources, while maintaining other optimization objectives. In T16 problem, all algorithms able to converge to the best solution, but the

ACO has better performance in term of the consistency. For this problem, ACO able to reach the optimum solution for every optimization run.

Table 4. Small size problem comparison

Test Problem	Algorithm	Minimum fitness	Maximum fitness	Average fitness	Standard deviation	f_1	f_2	f_3	f_4
T4	ACO	0.5505	0.5505	0.5505	0.0000	1	2	7	4
	GA	0.5505	0.5505	0.5505	0.0000	1	2	7	4
	PSO	0.5505	0.5505	0.5505	0.0000	1	2	7	4
	MPSO	0.5505	0.5505	0.5505	0.0000	1	2	7	4
T9	ACO	0.3094	0.3094	0.3094	0.0000	2	4	3	8
	GA	0.3094	0.3094	0.3094	0.0000	2	4	3	8
	PSO	0.3094	0.3094	0.3094	0.0000	2	4	3	8
	MPSO	0.3094	0.3094	0.3094	0.0000	2	4	3	8
T12	ACO	0.2531	**0.2531**	0.2531	**0.0000**	2	4	3	11
	GA	0.2531	**0.2531**	0.2531	**0.0000**	2	4	3	11
	PSO	0.2531	0.4380	0.3051	0.0733	2	4	3	11
	MPSO	**0.2455**	**0.2531**	**0.2470**	0.0031	2	4	3	10
T16	ACO	0.2151	**0.2151**	**0.2151**	**0.0000**	2	4	6	12
	GA	0.2151	0.4710	0.2506	0.0873	2	4	6	12
	PSO	0.2151	0.5076	0.4099	0.1065	2	4	6	12
	MPSO	0.2151	0.4068	0.2343	0.0590	2	4	6	12

The result of medium size problem in **Table 5** indicated that the MPSO and ACO lead the performance of the algorithm. The MPSO reached to minimum fitness and minimum average fitness in three cases. In the meantime, the ACO found the minimum fitness in two cases, while the minimum average in only one case. In T24, all algorithms able to search for minimum fitness, but again the ACO had better consistency. In T47 and T65 problems, the MPSO dominates the best minimum and average fitness compared with other algorithms. Meanwhile, in T70, the was ACO able to search for better minimum fitness, but the proposed MPSO had better average fitness and standard deviation.

Table 5. Medium size problem comparison

Test Problem	Algorithm	Minimum fitness	Maximum fitness	Average fitness	Standard deviation	f_1	f_2	f_3	f_4
T24	ACO	0.1899	**0.1930**	**0.1920**	**0.0011**	2	4	4	26
	GA	0.1899	0.1970	0.1927	0.0025	2	4	4	26
	PSO	0.1899	0.2144	0.2023	0.0096	2	4	4	26
	MPSO	**0.1899**	0.2073	0.1925	0.0053	2	4	4	26
T47	ACO	0.2697	0.3186	0.2989	0.0216	5	9	11702	88
	GA	0.3031	0.3270	0.3164	0.0079	5	10	13415	90
	PSO	0.2973	0.3231	0.3059	0.0077	5	10	10945	95
	MPSO	**0.1731**	**0.1776**	**0.1753**	**0.0020**	**4**	**8**	**2237**	**73**
T65	ACO	0.2586	0.2718	0.2674	**0.0041**	6	12	565	118
	GA	0.2612	0.3857	0.3332	0.0566	6	12	649	**113**
	PSO	0.2524	0.3822	0.2725	0.0388	6	12	469	**113**
	MPSO	**0.2491**	**0.2603**	**0.2569**	0.0052	**6**	**12**	**385**	116
T70	ACO	**0.4230**	0.4432	0.4339	0.0052	6	10	**273**	**97**
	GA	0.5492	0.6939	0.6205	0.0577	6	11	646	106
	PSO	0.4277	0.4556	0.4379	0.0096	6	10	303	99
	MPSO	0.4260	**0.4393**	**0.4316**	**0.0048**	6	10	293	98

Table 6 presents the optimization result for the large size problem. For this class of problem, the MPSO consistently able to search for better minimum fitness compared with comparison algorithms. Interm of average fitness, the MPSO had a better average in three cases, while the ACO had a better average in the remaining one case. The MPSO consistently found the minimum mated workstation, number of workstation and idle time in all cases of the large size problem.

Table 6. Large size problem comparison

Test Problem	Algorithm	Minimum fitness	Maximum fitness	Average fitness	Standard deviation	*f₁*	*f₂*	*f₃*	*f₄*
T83	ACO	0.4420	**0.4497**	0.4472	0.0032	6	11	39716	109
	GA	0.4886	0.4917	0.4906	**0.0014**	6	12	47593	118
	PSO	0.4329	0.4951	0.4598	0.0309	6	11	37109	107
	MPSO	**0.4324**	0.4524	**0.4400**	0.0079	**6**	**11**	**36944**	**107**
T111	ACO	0.3931	0.4206	0.4115	**0.0109**	7	13	67520	144
	GA	0.3212	0.4126	0.3737	0.0474	6	12	50709	136
	PSO	0.3177	0.4249	0.3952	0.0439	6	12	48681	135
	MPSO	**0.2989**	**0.3276**	**0.3200**	0.0120	**6**	**12**	**37365**	**135**
T148	ACO	0.2648	0.3682	**0.3070**	0.0553	5	10	565	119
	GA	0.2609	0.4228	0.3580	0.0865	5	10	515	118
	PSO	0.3623	0.4134	0.4003	**0.0214**	6	11	938	123
	MPSO	**0.2533**	**0.3608**	0.3092	0.0460	**5**	**10**	**405**	**122**
T205	ACO	0.2343	0.2399	0.2362	0.0023	5	10	4375	127
	GA	0.2363	0.3532	0.3236	0.0492	5	10	4575	126
	PSO	0.2343	0.3514	0.2990	0.0594	5	10	4375	127
	MPSO	**0.2308**	**0.2366**	**0.2348**	**0.0023**	**5**	**10**	**4055**	**126**

Next, a standard competition ranking method is used to analyze the results. In this approach, the algorithm with the best result will be assigned as rank 1, while the worst is rank 4. In the case the performance is tied, a similar rank will be given, and the next position will be left empty. **Table 7** present the frequency of the rank for every algorithm in term of minimum and average fitness.

Based on **Table 7**, the proposed MPSO was only ranked in rank 1 and rank 2 for both minimum and average fitness. For minimum fitness, the MPSO able to search for the best solution in 91.6% of the problems. At the same time, the MPSO was obtained better average fitness in 75% of the problems, while the remaining 25% in the second place. The MPSO also had a better average rank for minimum and average fitness. In both categories, the MPSO obtained 1.08 and 1.25 average rank respectively.

Table 7. Frequency of the rank for different algorithms

	Algorithm	Rank 1	Rank 2	Rank 3	Rank 4	Average Rank
Minimum fitness	ACO	5	3	3	1	2.00
	GA	4	2	1	5	2.58
	PSO	4	5	2	1	2.00
	MPSO	11	1	0	0	1.08
Average fitness	ACO	5	6	0	1	1.75
	GA	2	2	3	5	2.92
	PSO	2	0	6	4	3.00
	MPSO	9	3	0	0	1.25

The nearest challenger to MPSO is the ACO algorithm. The ACO had obtained the average rank 2.00 for minimum fitness, while 1.75 for average fitness. Meanwhile, the PSO algorithm also had the same average rank as ACO for minimum fitness, but in the last position for average fitness. It shows that the PSO converge to the different angle in the search space for the different optimization run. For this reason, the PSO come out with a different solution for the different run that made the fitness too diverse. In a different angle, this behavior has its own advantage, because the algorithm will explore a different side of the search space. However, it required a high number of repetition for the optimization run.

Figure 4 and **Figure 5** present the average rank by problem size for minimum and average fitness. In general, these figures show that for ACO, GA and PSO, the performance of the algorithm become worse when the problem size increased. This trend is related to the size of the search space. When the problem size increased, the number of possible solutions were excessively increased because of permutation combination. This made the searching process become harder and required for an efficient algorithm. In contrast, the MPSO is able to maintain the performance throughout the different problem size.

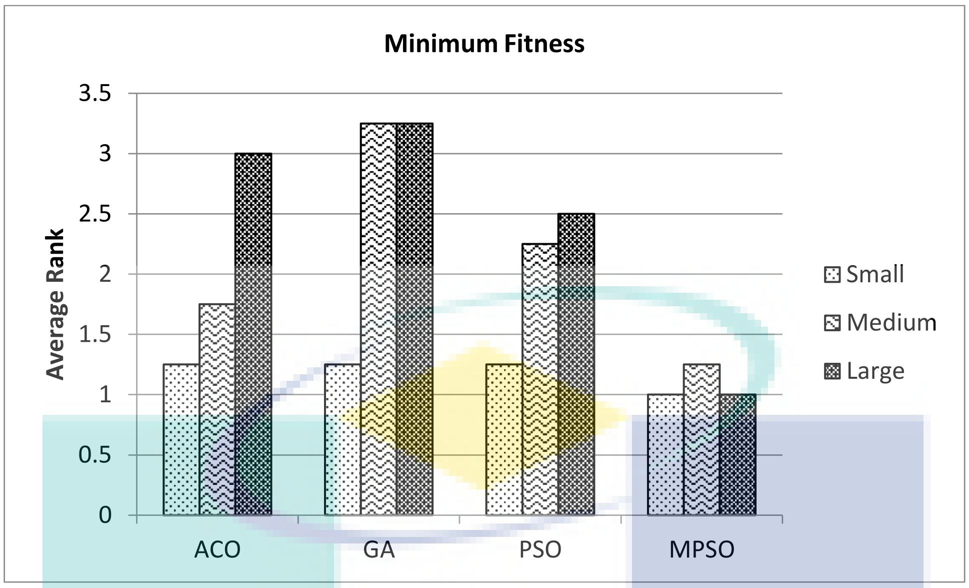


Figure 4. Minimum fitness by the problem size

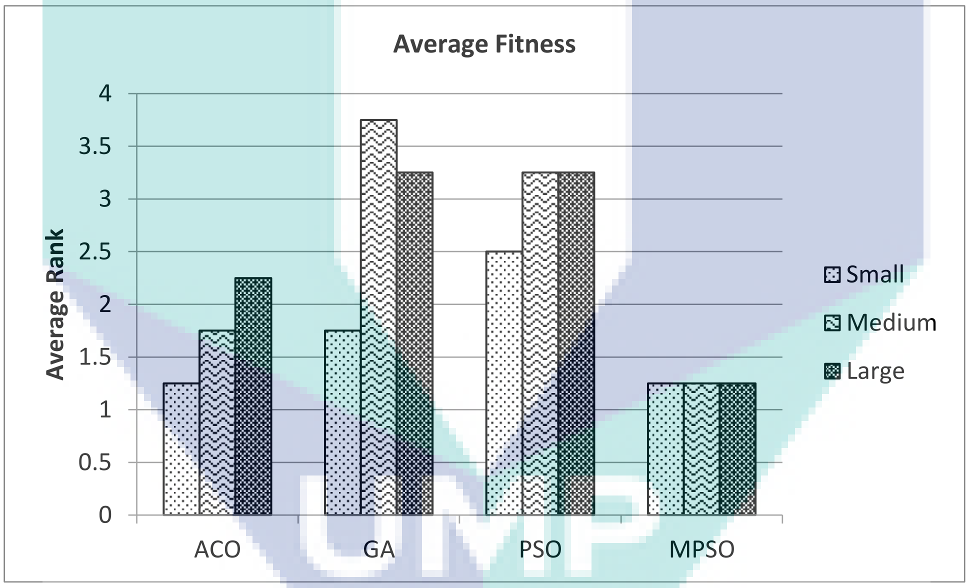


Figure 5. Average fitness by the problem size

Figure 6, **7** and **8** present the mean convergence for small, medium and large size problems respectively. For small size problem, the MMFO convergence was almost stagnant from iteration 180. Meanwhile in medium size problem, the MMFO convergence roughly stable at iteration 300. Then, a few small improvement still occurred until the end. For the large size problem, the convergence can be observed still occurred until the end of the run.

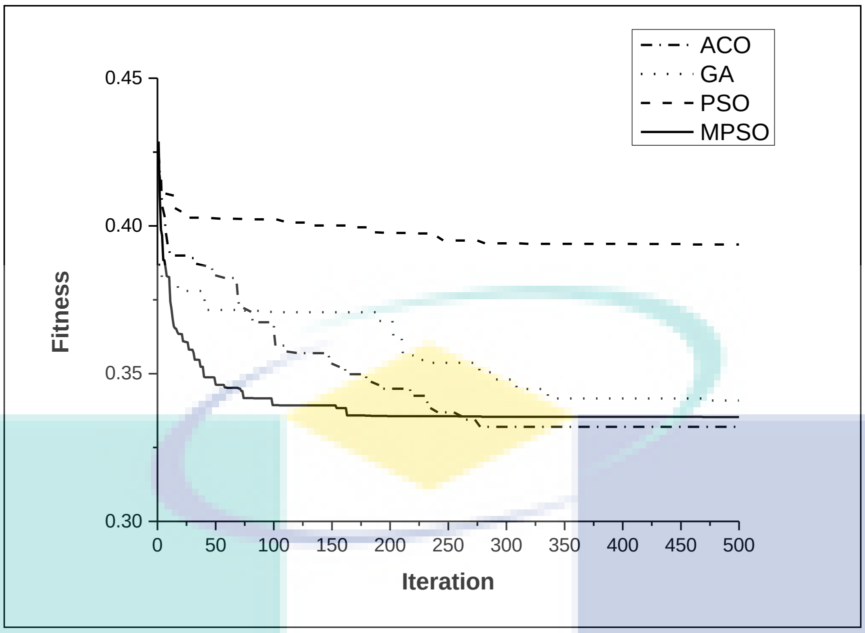


Figure 6. Convergence plot of small size problem

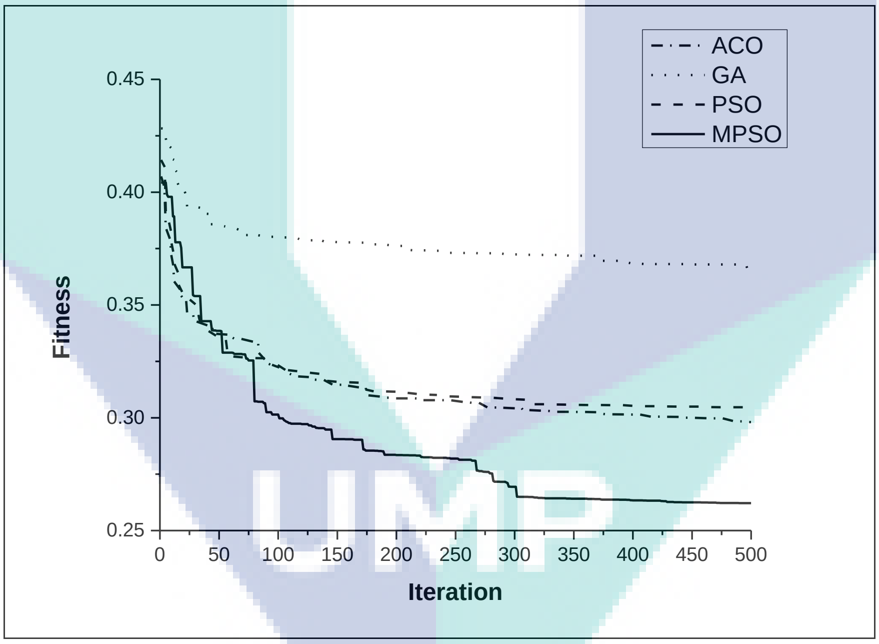


Figure 7. Convergence plot of medium size problem

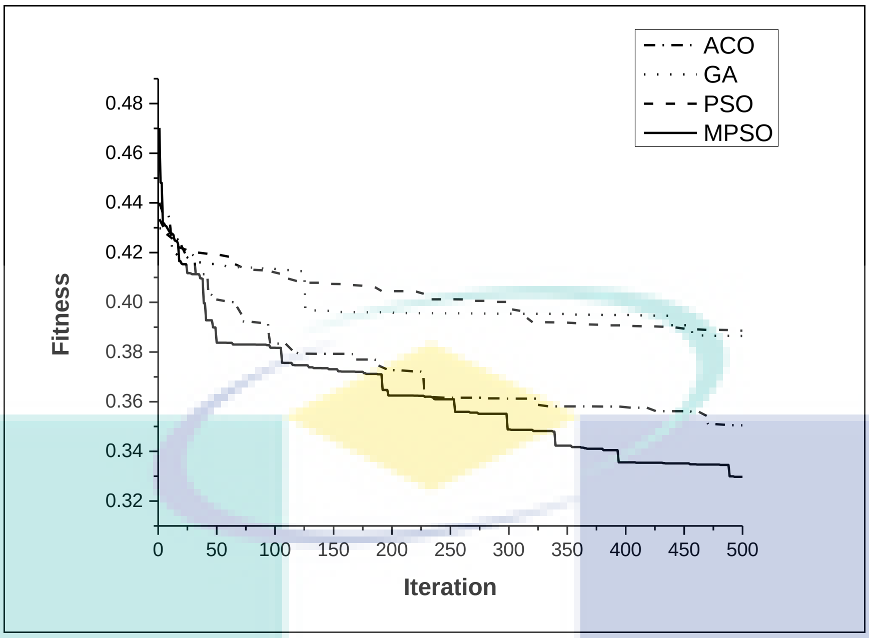


Figure 8. Convergence plot of large size problem

In small size problem, where the search space is also relatively small, the MMFO algorithm manages to converge faster. This can be observed from the steep slope for the first 75 iterations in **Figure 6**. On the other hand, the early MMFO convergence in medium size problem was intermixed between steep and short flat slopes. Meanwhile, the longer flat slope can be observed in large size problem with periodical steep slopes. The patterns of convergence in small, medium and large size problems were affected by the size of search space. When the problem size increased, the number of possible solution was also increased. Furthermore, in the smaller problem, tiny changes in the assembly sequence gave more effect on the fitness value compared with the larger problem because of the ratio between the changes and problem size.

5. Conclusion & Future Work

This paper modeled and optimized the two-sided assembly line balancing (2S-ALB) with resource constraints. In different with the majority of existing works that assume all workstations have similar capabilities, this research considers the assembly resources including tools, machines and workers to be minimized during the line balancing. For optimization purpose, Modified Particle

Swarm Optimization (MPSO) was introduced by considering top three solutions as the global best (Gbest) instead of one best solution in PSO algorithm. This changes made to maintain the solution diversity over the iterations.

A computational experiment was conducted by using 12 benchmark test problems from small, medium and large size. The optimization results of MPSO were compared with results from popular algorithms for 2S-ALB, including Genetic Algorithm (GA), Ant Colony Optimization (ACO) and PSO algorithms. The computational experiment results indicated that the proposed MPSO able to search for the best solution in 11 out of 12 test problems. Unlike the comparison algorithms, the MPSO is capable to maintain the performance when the problem size increased. Besides that, the results also indicated that the proposed model for 2S-ALB with resource constraints able to reduce the number of resources in an assembly line. This is important to set up the assembly line in an efficient way.

The modification on the Gbest has made the MPSO become more dynamic in terms of the search direction. This change has two-fold advantages. The first advantage is the proposed MPSO had better exploration that makes the chances to obtain an optimum solution is higher. Meanwhile, the second advantage is the possibility for the algorithm to trap in local optima can be reduced. This work however has a drawback in terms of the multi-objective handling. Since this work implemented the weighted sum approach for the multi-objective problem, the result is highly depended on the weight used for each optimization objective. Currently, a similar weight is assigned to all optimization objectives. In future, a study to determine a suitable weight for different optimization objectives is proposed. Finally, the Pareto optimality concept for multi-objective handling is suggested to have a better view on the optimum solution.

Acknowledgement

The authors would like to acknowledge the Ministry of Higher Education, Malaysia and Universiti Malaysia Pahang for supporting this research under the FRGS grant RDU140103.

References

[1] M. H. Alavidoost, M. Tarimoradi, and M. H. F. Zarandi, "Fuzzy adaptive genetic algorithm for multi-objective assembly line balancing problems," *Appl. Soft Comput. J.*, vol. 34, pp. 655–677, 2015.

[2] M. R. A. Make, M. F. F. Rashid, and M. M. Razali, "Modelling of Two-sided Assembly Line Balancing Problem with Resource Constraints," in *IOP Conference Series: Materials Science and Engineering*, 2016, vol. 160, no. 1.

[3] G. Tuncel and D. Aydin, "Two-sided assembly line balancing using teaching-learning based optimization algorithm," *Comput. Ind. Eng.*, vol. 74, no. 1, pp. 291–299, 2014.

[4] J. J. Bartholdi, "Balancing two-sided assembly lines: A case study," *Int. J. Prod. Res.*, vol. 31, no. 10, pp. 2447–2461, 1993.

[5] Y. K. Kim, W. S. Song, and J. H. Kim, "A mathematical model and a genetic algorithm for two-sided assembly line balancing," *Comput. Oper. Res.*, vol. 36, no. 3, pp. 853–865, 2009.

[6] H. D. Purnomo and H.-M. Wee, "Maximizing production rate and workload balancing in a two-sided assembly line using Harmony Search," *Comput. Ind. Eng.*, vol. 76, no. 1, pp. 222–230, 2014.

[7] P. Chutima and W. Naruemitwong, "A Pareto biogeography-based optimisation for multi-objective two-sided assembly line sequencing problems with a learning effect," *Comput. Ind. Eng.*, vol. 69, no. 1, pp. 89–104, 2014.

[8] D. Khorasanian, S. R. Hejazi, and G. Moslehi, "Two-sided assembly line balancing considering the relationships between tasks," *Comput. Ind. Eng.*, vol. 66, no. 4, pp. 1096–1105, 2013.

[9] B. Yuan, C. Zhang, X. Shao, and Z. Jiang, "An effective hybrid honey bee mating optimization algorithm for balancing mixed-model two-sided assembly lines," *Comput. Oper. Res.*, vol. 53, pp. 32–41, Jan. 2015.

[10] P. Chutima and P. Chimklai, “Multi-objective two-sided mixed-model assembly line balancing using particle swarm optimisation with negative knowledge,” *Comput. Ind. Eng.*, vol. 62, no. 1, pp. 39–55, 2012.

[11] A. S. Simaria and P. M. Vilarinho, “2-ANTBAL: An ant colony optimisation algorithm for balancing two-sided assembly lines,” *Comput. Ind. Eng.*, vol. 56, no. 2, pp. 489–506, Mar. 2009.

[12] M. R. Abdullah Make, M. F. F. Ab. Rashid, and M. M. Razali, “A review of two-sided assembly line balancing problem,” *Int. J. Adv. Manuf. Technol.*, vol. 89, no. 5–8, pp. 1743–1763, Aug. 2017.

[13] P. Tapkan, L. Özbakir, and A. Baykasoılu, “Bee algorithms for parallel two-sided assembly line balancing problem with walking times,” *Appl. Soft Comput. J.*, vol. 39, pp. 275–291, 2016.

[14] I. Kucukkoc and D. Z. Zhang, “Type-E parallel two-sided assembly line balancing problem: Mathematical model and ant colony optimisation based approach with optimised parameters,” *Comput. Ind. Eng.*, vol. 84, pp. 56–69, 2015.

[15] I. Kucukkoc and D. Z. Zhang, “A mathematical model and genetic algorithm-based approach for parallel two-sided assembly line balancing problem,” *Prod. Plan. Control*, vol. 26, no. 11, pp. 874–894, 2015.

[16] U. Özcan, H. Gökcen, and B. Toklu, “Balancing parallel two-sided assembly lines,” *Int. J. Prod. Res.*, vol. 48, no. 16, pp. 4767–4784, 2010.

[17] Y. K. Kim, Y. Kim, and Y. J. Kim, “Two-sided assembly line balancing: A genetic algorithm approach,” *Prod. Plan. Control*, vol. 11, no. 1, pp. 44–53, Jan. 2000.

[18] T. O. Lee, Y. Kim, and Y. K. Kim, “Two-sided assembly line balancing to maximize work relatedness and slackness,” *Comput. Ind. Eng.*, vol. 40, no. 3, pp. 273–292, 2001.

[19] Y. Delice, E. Kızılkaya Aydoğan, and U. Özcan, “Stochastic two-sided U-type assembly line balancing: a genetic algorithm approach,” *Int. J. Prod. Res.*, vol. 54, no. 11, pp. 3429–3451,

2016.

[20] R. B. Taha, A. K. El-Kharbotly, Y. M. Sadek, and N. H. Afia, “A Genetic Algorithm for solving two-sided assembly line balancing problems,” *Ain Shams Eng. J.*, vol. 2, no. 3–4, pp. 227–240, 2011.

[21] A. Baykasoglu and T. Dereli, “Two-sided assembly line balancing using an ant-colony-based heuristic,” *Int. J. Adv. Manuf. Technol.*, vol. 36, no. 5–6, pp. 582–588, 2008.

[22] I. Kucukkoc and D. Z. Zhang, “Mixed-model parallel two-sided assembly line balancing problem: A flexible agent-based ant colony optimization approach,” *Comput. Ind. Eng.*, vol. 97, pp. 58–72, 2016.

[23] Z. Zhang, J. Hu, and W. Cheng, “An ant colony algorithm for two-sided assembly line balancing problem type-II,” *Adv. Intell. Syst. Comput.*, vol. 213, pp. 369–378, 2014.

[24] X. Hu, E. Wu, and Y. Jin, “A station-oriented enumerative algorithm for two-sided assembly line balancing,” *Eur. J. Oper. Res.*, vol. 186, no. 1, pp. 435–440, 2008.

[25] P. Fattahi, P. Samouei, and M. Zandieh, “Simultaneous Multi-skilled Worker Assignment and Mixed-model Two-sided Assembly Line Balancing,” *Int. J. Eng.*, vol. 29, no. 2, pp. 211–221, 2016.

[26] W. Chiang, T. L. Urban, and C. Luo, “Balancing stochastic two-sided assembly lines,” *Int. J. Prod. Res.*, vol. 54, no. 20, pp. 6232–6250, 2016.

[27] Y. Delice, E. Kızılkaya Aydoğan, U. Özcan, and M. S. İlkay, “A modified particle swarm optimization algorithm to mixed-model two-sided assembly line balancing,” *J. Intell. Manuf.*, vol. 28, no. 1, pp. 23–36, 2017.

[28] Y. Delice, E. K. Aydoğan, U. Özcan, and M. S. İlkay, “Balancing two-sided U-type assembly lines using modified particle swarm optimization algorithm,” *4or*, 2016.

[29] Z. Li, M. N. Janardhanan, Q. Tang, and P. Nielsen, “Co-evolutionary particle swarm

optimization algorithm for two-sided robotic assembly line balancing problem," *Adv. Mech. Eng.*, vol. 8, no. 9, pp. 1–14, 2016.

[30] Q. Tang, Z. Li, L. Zhang, and C. A. Floudas, "A hybrid particle swarm optimization algorithm for large-sized two-sided assembly line balancing problem," *ICIC Express Lett.*, vol. 8, no. 7, pp. 1981–1986, 2014.

[31] H. D. Purnomo, H. Wee, H. Rau, H. Dwi, H. Wee, and H. Rau, "Two-sided assembly lines balancing with assignment restrictions," *Math. Comput. Model.*, vol. 57, no. 1–2, pp. 189–199, Jan. 2013.

[32] M. A. Adnan and M. A. Razzaque, "A comparative study of Particle Swarm Optimization and Cuckoo Search techniques through problem-specific distance function," in *2013 International Conference of Information and Communication Technology, ICoICT 2013*, 2013, pp. 88–92.

[33] O. Rubiano-Ovalle and A. Arroyo-Almanza, "Solving a two-sided assembly line balancing problem using memetic algorithms," *Ing. y Univ.*, vol. 13, no. 2, pp. 267–280, 2009.

[34] A. Scholl, "Benchmark Data Sets by Scholl," *Assembly Line Balancing Data Dets & Research Topics*, 1993. [Online]. Available: http://assembly-line-balancing.mansci.de/salbp/benchmark-data-sets-1993/.

[35] M. F. F. Rashid, W. Hutabarat, and A. Tiwari, "A review on assembly sequence planning and assembly line balancing optimisation using soft computing approaches," *Int. J. Adv. Manuf. Technol.*, vol. 59, no. 1–4, pp. 335–349, Aug. 2011.

CHAPTER 5

CONCLUSIONS

This chapter summarises the research works and highlights the contribution of the research to knowledge. Lastly, some recommendations for the future research are suggested.

5.1 SUMMARY OF RESEARCH

This section summarises the work that had been done throughout the research. The first phase in this research is conductance of literature review. The literature review presented the classification of assembly line balancing. The literature review focus on the problem modelling for simple ALB type E (ALB-E), two-sided ALB (2S-ALB) and mixed-model ALB (MMALB). Besides identify the problem modelling, the optimization objectives, algorithms and benchmark test problem were also identified in the literature.

Next, the ALB with resource constraints was modelled in mathematical form. This phase includes how to transform the assembled product into mathematical form. Then proposed suitable mathematical model and constraints to deal with combinatorial problems. Besides that, this phase also involved the objective function formulation. In different ALB variation, sometimes the different objective function were used. For example in MMALB, one of the optimization objective is to minimize production rate variation (PRV) that related with the sequence of different model to be assembled. Meanwhile, in 2S-ALB, the number of mated workstation is important to minimize the assembly space in the plant.

Computational test were conducted to separately for ALB-E, 2S-ALB and MMALB to test the performance of the optimization algorithm. The computational test were conducted using standard benchmark test problems. In some cases, the algorithm need to be modified to meet the optimum solutions. In 2S-ALB for instance due to the early convergence of PSO, this algorithm were modified by adding top three solutions instead of a single best for the reproduction purpose.

Apart from that, an industrial case study was conducted in BI Technologies Sdn. Bhd. to validate the applicability of optimised algorithm as well as the mathematical model. The related assembly data had been collected to be simulated in WITNESS™ Simulation Software. The existing layout simulation is used to validate the simulation model with actual layout. The feedback from the industrial expert indicated that the output from the optimised algorithm cannot be fully implemented due to the fact that their equipment is isolated and cannot be moved to another place. Nevertheless, the industrial expert concluded that the proposed algorithm and model is applicable and can be implemented for industrial application as it can minimise the usage of resources and the number of workstation in production line. In fact, it also can enhance the line efficiency by minimising production time.

5.2 CONCLUSIONS

Based on the research that was conducted, there were a few important points to be highlighted. The first one is the problem modelling for ALB with resource constraints can be generalized for different ALB problems. In this context, the simple ALB type E (ALB-E), two-sided ALB (2S-ALB) and mixed-model ALB (MMALB) can be generalized for the resource constraints. However, the fundamental problem formulation must be made separately. The establishment of the ALB with resource constraints model was a clear indicator of the achievement for the first research objective.

The second conclusion is regarding the optimization algorithm that related with the second research objective. In this context, no single algorithm that has better performance in all ALB problem type. In 2S-ALB, the Particle Swarm Optimization (PSO) performed better, while in MMALB, the Ant Colony Optimization (ACO) algorithm has better solution. This finding was also in line with the "No Free Lunch Theorems" that no single algorithm is suitable for all problems.

Next, the industrial case study results indicated that the proposed ALB with resource constraints is able to reduce the number of resources, while maintaining the capability to balance the assembly line. It confirmed that the proposed model is capable to be implemented in real industry. The result from case study also conclude the third research objective.

5.3 LIMITATION AND RECOMMENDATION

Even though the objectives of this research were successfully achieved, there were a few limitations that need to be point out. The first limitation is related with the software and machine capability. In this research, the optimization algorithms were programmed in MATLAB. Since MATLAB is for the prototype development, it requires longer CPU time. Furthermore, the limitation of the CPU also contributed to the high CPU time. Therefore, in this research, the optimization is only run for 500 iteration, considering the time limit that the researchers have.

The second limitation is the algorithm comparison was limited to robust and popular metaheuristic algorithms (i.e. GA, PSO and ACO). Recently, there were various new optimization algorithms that proposed by researchers for different application. These algorithms have the potential to be explored for the line balancing problem.

Finally, the research limitation is in term of the application of the results in industry. In this research, the industrial case study was conducted by collecting the data, optimize using the proposed model and simulate using the discrete event simulation tool. The output later was validated by the expert from industry. The implementation of the result in real assembly line will enhanced the assembly efficiency.