Assembly line balancing with resource constraints using new rank-based crossovers

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.

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.

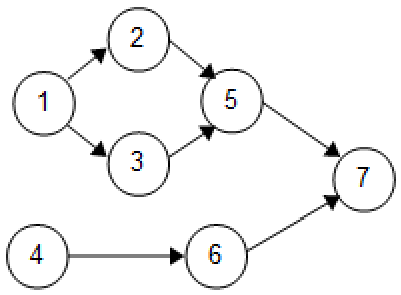


Figure 1. Example of precedence graph

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, $\mathrm{ct}_{\max}$ is 34 time unit. It means that the workstation time cannot exceed the $\mathrm{ct}_{\max}$ 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.

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

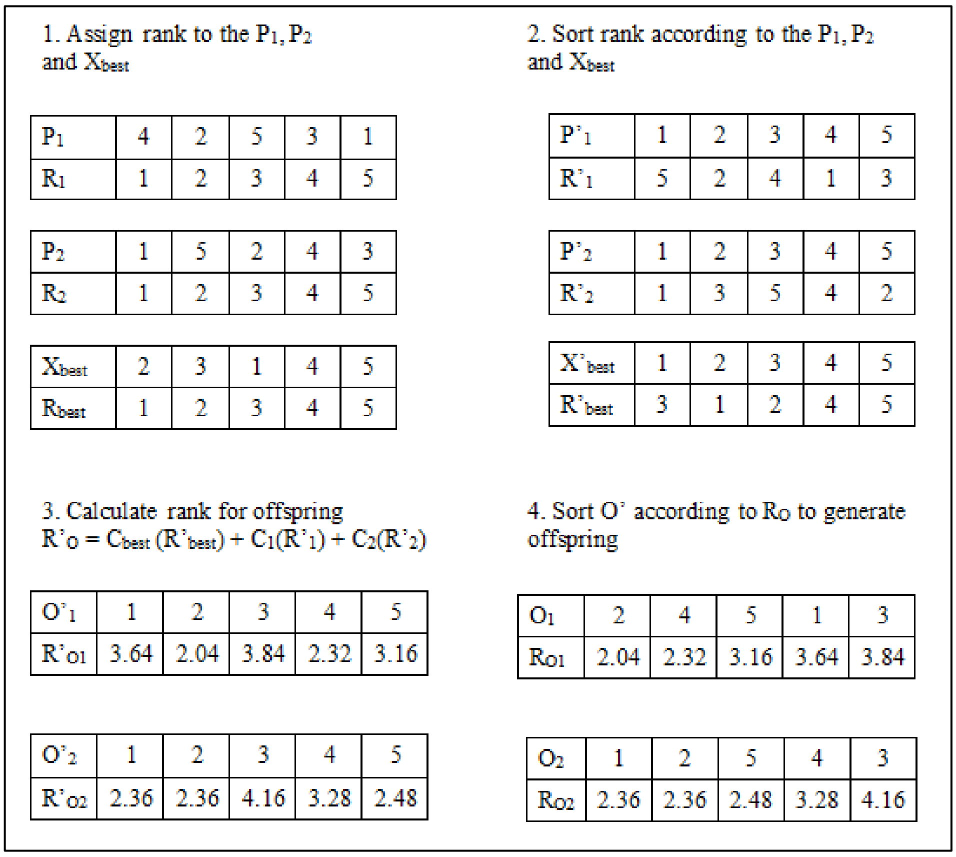
$$C_2 = 0.3(1 - C_{best}) \qquad (3)$$

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.

6. Acknowledgments

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

7. References

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

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

[3] Hamta N, Fatemi-Ghomi S M T, Jolai F, and Shirazi M A Jan.2013 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

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

[5] Corominas A, Ferrer L, and Pastor R Jun. 2011Assembly line balancing: General resource-constrained case *Int. J. Prod. Res.*, **vol. 49** no. 12, pp. 3527–42

[6] Koltai T and Tatay V 2013 Formulation of workforce skill constraints in assembly line balancing models *Optim. Eng.*, **vol. 14** no. 4, pp. 529–45

[7] Jusop M and Rashid M F F A 2016 Optimisation of assembly line balancing type-E with resource constraints using NSGA-II *Key Eng. Mater.* **vol. 701** pp. 195–99

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

[9] Triki H, Mellouli A, and Masmoudi F 2017 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

[10] Jusop M and Rashid M 2017 Optimization of Assembly Line Balancing with Resource Constraint using NSGA-II : A Case Study *Int. J. Appl. Eng. Res.*, **vol. 12** no. 7 pp. 1421–26

[11] Delice Y, Aydoğan E K, and Özcan U 2016 Stochastic two-sided U-type assembly line balancing: a genetic algorithm approach *Int. J. Prod. Res.*, **vol. 54** no. 11 pp. 3429–51

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

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