OPEN ACCESS
Saudi Journal of Engineering and Technology
Abbreviated Key Title: Saudi J Eng Technol
ISSN 2415-6272 (Print) |ISSN 2415-6264 (Online)
Scholars Middle East Publishers, Dubai, United Arab Emirates
Journal homepage: http://scholarsmepub.com/sjet/

**Original Research Article**

# Single Model Assembly Line Balancing Using Enhanced Genetic Algorithm

Shady Magdy Anwar*, Ahmed Mahmoud Ali, Mohamed Ahmed Awad

Faculty of Engineering, Ain Shams University, Abdo Pasha Square, Cairo, Egypt

**DOI:** 10.36348/sjet.2019.v04i12.003 | **Received:** 16.12.2019 | **Accepted:** 23.12.2019 | **Published:** 25.12.2019

***Corresponding author:** Shady Magdy Anwar
**Email:** shadymagdy@windowslive.com

## Abstract

Assembly line balancing (ALB) is used in many industries to minimize the number of stations, improve the efficiency and work load balance among stations. Enhanced Genetic Algorithm (EGA) is proposed using precedence preservative crossover and scramble mutation techniques, aiming to minimize two objectives; number of stations as a primary objective and smoothing index as a secondary objective. Benchmark problems were selected from the literature, used to test the efficacy of the algorithm and to compare the results with the well-known algorithm SALOME. The results showed high efficacy while handling two objectives. It outperformed SALOME in work load balance and efficiency. On the other, EGA outperformed the Genetic Algorithm (GA) developed by [1] in minimizing the number of stations.

**Keywords:** Assembly line balancing, Genetic Algorithms, SALOME, simple assembly line balancing problem, precedence preservative crossover, Multi-objective optimization.



## INTRODUCTION

Assembly lines are widely used in industry to group work elements in workstations with a pre-specified cycle time and perform the operations without violating the cycle time and precedence relations. Computationally, ALB is considered as NP-hard problems. Therefore, it will be time consuming to search the optimal solution for real life problems. Exhaustive enumerations-needed to reach such optimal-directed the researchers towards heuristic and meta-heuristic methods.

GA is used nowadays to find the most promising solutions for combinatorial optimization problems, where the optimal solution cannot be found within a reasonable time.

Many approaches used to solve ALB problem using heuristic and meta-heuristic methods [1]. Was the first to apply genetic algorithms to solve ALB problem using two-point crossover and scramble mutation, with two objectives, namely, minimizing smoothing index (SI) and number of stations [2]. Applied local search method with genetic algorithms to minimize the cycle time and applied different crossover operators to compare between them [3]. Tested different crossover and mutation operators in solving different single-objective and multi objective problems. Sabuncuoglu, I *et al*., [4] introduced a novel technique called dynamic partitioning by freezing the first and last stations during the evolution process, he also applied simulated annealing for the fittest member of population.

A multi-objective algorithm based on simulated annealing was developed by [5] to solve SALBP and U-type assembly lines. The first objective is maximizing efficiency and the second objective is minimizing the smoothing index and constructing feasible solutions based on task assignment priority rules. The proposed algorithm was tested on bench mark problems and the optimal solution was found for all the tested problems.

A multi objective simulated annealing algorithm proposed by [6] to minimize design costs and smoothing index. A new multinomial probability mass function was introduced for the acceptance of worst solutions. The obtained results Pareto based help the decision maker to evaluate a great number of alternatives.

A particle swarm optimization algorithm developed by [7], based on bird flocking and fish school theories to solve ALBP with two objectives; the first is maximization of production rate by minimizing the cycle time and the second is maximizing workload smoothness. The performance of the algorithm is tested

on some of benchmark problems and the results showed high performance for the proposed approach.

[8] applied cellular rearranging techniques to solve ALB problem with two main objectives: minimizing number of stations and minimizing smoothing index.

From the above brief literature we can see that few researchers, while solving ALB, they used weighted objective functions. I.e. they give actually large weight or the same weight to the smoothing index and small weight to the number of stations.

In this paper, an Enhanced GA (EGA) is proposed to minimize number of stations as a first objective and smoothing index as a second objective. The algorithm was applied to the benchmark problems provided by [9] and compared with optimal solutions obtained by the well-known method SALOME [9].

The paper is organized as follows: in section 2, the proposed study. Section 3, Results and discussion. Section 4, conclusion and recommendations.

## THE PROPOSED STUDY

This section describes the problem studied and the methodology of the developed algorithm.

### Problem Description

The simple assembly line balancing problem (SALBP) take up a single-model, straight line and one-sided assembly line with deterministic work-elements times. SALBP can be classified according to their objective function as the following:

Type-1: minimizing the number of work stations for a fixed cycle time.

Type-2: minimizing the cycle time for a fixed number of work stations

Type-E: this type concerning minimizing cycle time and number of stations to reach the highest efficiency.

Type-F: this type is associated to assure the feasibility of solution for a given cycle time and number of stations.

Type-3, 4 and 5: corresponds to minimize smoothing index, maximization work relatedness and multiple objectives with type-3 and type-4 respectively.

In this research, we will solve SALBP of type-1 problems, using the same assumptions given by [9].

### Objective Function

[10] stated that lower smoothing index (SI) is not indicator for a lower number of stations; moreover, using SI as objective function is not appropriate with SALBP-1 objectives and inferior solutions could be selected.

We used multi-objective optimization function to solve our problem. In the weighted function. A high weight is attributed to the number of stations (represented by balance delay (BD), calculated for predetermined cycle time) and small weight is attributed to SI (to break the ties).

Min. Objective function = 0.98×BD+0.02×SI

### Enhanced Genetic Algorithm (EGA)

Genetic algorithms, based on the evolution theory, are effective in large search spaces, non-linear, unknown spaces...etc.

Most researchers used two-point crossover technique; besides, nobody tried to hybridize non-traditional generation methods for population with heuristic solutions.

EGA makes some enhancements to the algorithm developed by [1]. EGA uses precedence preservative crossover technique [11] instead of two-point crossover; and scramble mutation.

In addition, five heuristic methods are used in the initial population with solutions obtained by population generation method introduced by [13].

The objective function used, attributed large weight to number of stations and small weight to SI. By way of contrast [1], attributed large weight to SI and small weight to the number of stations.

Concerning survival selection, we used fitness-based selection method, so as to have a population of the fittest solutions, while the selection method of parents for reproduction is based on the well-known roulette wheel selection [12].

### Initial Population

The first five chromosomes in the population generated are based on five heuristic methods (Rank positional weight, minimum work elements time, Minimum number of immediate followers, maximum work elements time and maximum number of immediate followers). These five methods solved using station oriented method as explained in Scholl [9].

The rest of population is divided into three thirds. Each of these three thirds has solutions generated by forward method, backward method and combined method as explained by [13].

### Precedence Preservative Crossover

This crossover operator [11], was not used before for solving SALBP, depends on generating random binary vector (0, 1). Two offsprings generated, for the first off spring, if the gene value equal to (0) then the genes of the new offspring are selected from the first parent, if the gene value equal to (1) then genes selected from the second parent. For the second

offspring, if the gene value equal to (0) then the genes of the new offspring are selected from the second parent, if the gene value equal to (1) then genes selected from the first parent. Fig-1 shows an example of randomly generated binary vector [0 1 0 0 1 0 1 0 1], the steps of this operator are as follows:

- First gene in the binary vector is (0), then choose the first gene for first offspring from first parent and delete the gene from parent-2 to avoid choosing it second time.
- Second gene value equal (1), then choose the second gene from parent-2 and remove the gene from parent-1.
- Repeat the above steps till the offspring is filled with all work elements.
- Repeat the above steps for offspring-2 by replacing parent-1 by parent-2 and vice versa.

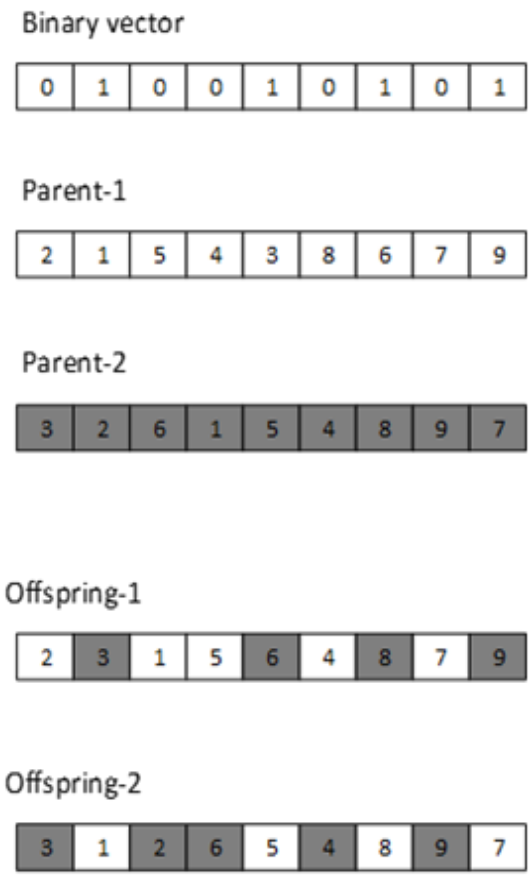


**Fig-1: Precedence preservative Crossover example**

**Scramble Mutation**

In GA, Crossover operator is not sufficient for optimization, crossover may prevent the exploration of new areas in the solution space. Mutation process can help the algorithms to escape the local optimal and reach the global optimum. In our research scramble mutation is used and applied to the selected parents. Fig-2 shows an example for scramble mutation, the steps are as follows:

- A random point is selected.
- All points before the selected point added to the new off-spring without change.
- All points after the selected point reconstructed randomly without violating the precedence constraints.

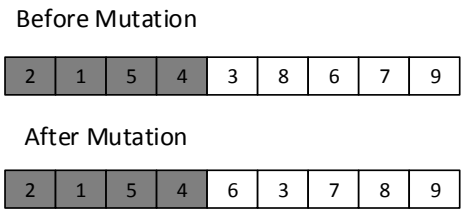


**Fig-2: Scramble mutation example**

**EGA steps**

EGA steps, shown in Fig-3 are as follows:

1. Select the number of iterations, population, crossover (Pc) and mutation (Pm) probabilities.
2. Select two parents using the well-known method roulette wheel selection.
3. Generate two random numbers between 0 and 1 for crossover and mutation probabilities.
4. If random number is less than or equal the crossover probability, proceed to crossover.
5. If random number is less than or equal mutation probability, proceed to mutation.
6. If the random numbers are greater than the crossover or mutation probabilities, respectively, no crossover or mutation, go to the next iteration.
7. Assign work elements of the obtained solution from crossover and mutation to work stations, based on station oriented method [9] and calculate fitness function for each solution.
8. Each offspring has fitness function higher than that of parents replaces the worst chromosome in the population.
9. Repeat steps from 2 to 8 till the stopping criteria is reached.

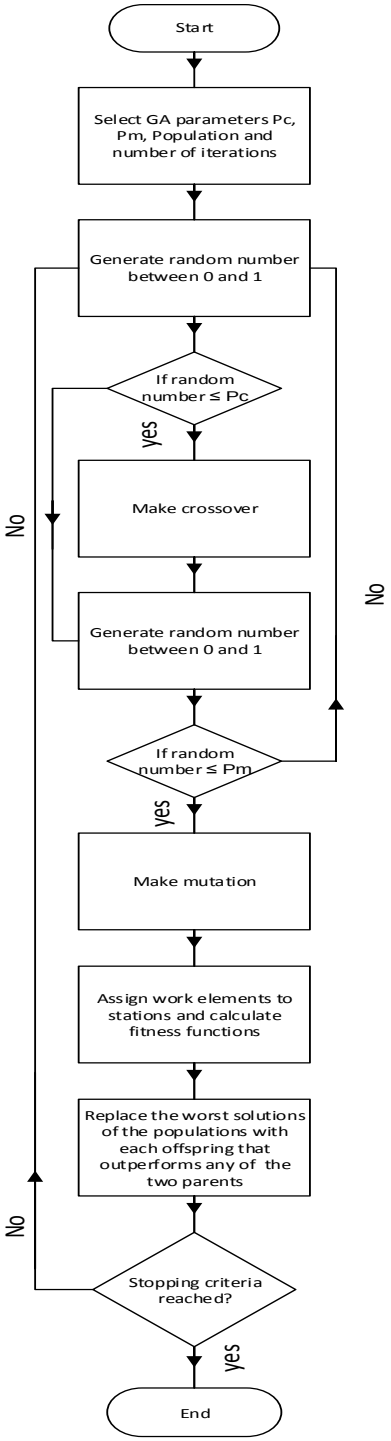


**Fig-3: Flow chart of EGA**

**Fitness Function**

Genetic algorithm is based on surviving the fittest concept while roulette wheel selection is based on the probability of selection according to highest probability of fitness function. What is more, our objective function is minimizing function. Thus, minimizing the objective function is equivalent to maximizing the inverse of the function as shown below:

Fitness function= 1/Objective function.

**EGA Parameters**

Based on experiments done to calibrate the proposed algorithm, the genetic parameters are as follows:

Number of iterations: 500.
Number of population: 25.
Crossover probability: 95%.
Mutation probability: 2%

## RESULTS AND DISCUSSION

In order to test the performance of the proposed algorithm and to compare with the well-known algorithm SALOME, the proposed procedure was coded using C++.

In Table-1 only 30 optimal solutions were selected-out of 171 benchmark solutions [9]-in this paper, for the reason of comparison with SALOME. Three performance criteria are used to evaluate the algorithm (number of stations (n), line efficiency (E) and smoothing index (SI)).

For the set of Jackson, the problem was solved with six different cycle times. For cycle times 7, 9 and 10, both algorithms yielded the same efficiency. On the other hand, EGA outperformed SALOME from SI perspective at those cycle times. At cycle times 13, 14 and 21, EGA outperformed SALOME from efficiency and SI perspectives, respectively.

For small problems, at small cycle times, EGA becomes more restricted and cannot find better station loads with lower cycle times. Thus, EGA yield same efficiency as SALOME. As the cycle time increases, EGA can find more assignment alternatives with better efficiency and SI.

**Table-1: The results of solved benchmark problems**

| Prob. ID | Cycle time | n[1] | SALOME | | EGA | |
|---|---|---|---|---|---|---|
| | | | E[2] (%) | SI[3] | E (%) | SI |
| Jackson (11 work elements) | 7 | 8 | 82.1 | 5.5 | 82.1 | 4.7 |
| | 9 | 6 | 85.2 | 4.7 | 85.2 | 4.2 |
| | 10 | 5 | 92 | 3.2 | 92 | 2.4 |
| | 13 | 4 | 87 | 4.5 | 95.8 | 1.4 |
| | 14 | 4 | 78 | 8.2 | 95.8 | 1.4 |
| | 21 | 3 | 63 | 16 | 80.7 | 10 |
| Roszieg (25 work elements) | 14 | 10 | 89.29 | 7.81 | 89.29 | 7.68 |
| | 16 | 8 | 97.66 | 2.23 | 97.66 | 2.23 |
| | 18 | 8 | 86.8 | 15.26 | 91.91 | 5 |
| | 21 | 6 | 99.2 | 1 | 99.2 | 1 |
| | 25 | 6 | 83.3 | 18.68 | 90.57 | 7 |
| | 32 | 4 | 97.65 | 2.23 | 97.65 | 2.23 |
| Hahn (53 work elements) | 2004 | 8 | 89.09 | 996 | 90.09 | 950 |
| | 2338 | 7 | 85.78 | 1762 | 85.78 | 1525 |
| | 2806 | 6 | 83.3 | 2396 | 85.5 | 1242 |
| | 3507 | 5 | 80.24 | 2934 | 80.24 | 1993 |
| | 4676 | 4 | 75.3 | 4676 | 75.3 | 3641 |
| Tonge (70 work elements) | 160 | 23 | 95.38 | 48.29 | 96.58 | 33.7 |
| | 168 | 22 | 94.97 | 51.22 | 95.53 | 42.04 |
| | 185 | 20 | 94.86 | 79.78 | 95.38 | 48.33 |
| | 207 | 18 | 94.2 | 117.82 | 95.12 | 53.68 |
| | 270 | 14 | 92.86 | 194.3 | 96.42 | 44.15 |
| | 410 | 9 | 95.12 | 120.92 | 98.48 | 22.04 |
| Mukhereje (94 work elements) | 176 | 25 | 95.64 | 68.65 | 96.18 | 46.05 |
| | 183 | 24 | 95.81 | 76.45 | 96.33 | 51.8 |
| | 192 | 23 | 95.3 | 79.86 | 95.78 | 77.73 |
| | 211 | 21 | 95 | 99.46 | 95.87 | 93.7 |
| | 234 | 19 | 94.7 | 152.25 | 95.87 | 59.01 |
| | 248 | 18 | 94.3 | 138.74 | 95.81 | 75.52 |
| | 263 | 17 | 94.12 | 161.32 | 95.20 | 67.3 |

[1] Number of stations.
[2] Line efficiency.
[3] Smoothing Index.

The rationale behind that is the dominance rules used in SALOME; as Jackson dominance rule and maximum load rule, tends to build fully loaded stations at the start of the line and stations with large idle times at the end of the line. Hence, work elements are not distributed with equity among stations. When in fact, EGA can reach better search spaces that SALOME cannot reach.

In medium problems, as Roszieg, the problem was solved with six different cycle times. For cycle times 14, 16, 21 and 32, both algorithms yielded the same efficiency and smoothing index. For other cycle times, EGA outperformed SALOME from efficiency and SI perspectives.

At small cycle times, like 14 in Roszieg set, assignment alternatives decrease. Thus, EGA cannot find better station loads. Even more, at cycle times 16, 21 and 32 the stations was quasi fully-loaded, in that case, EGA and cannot reach better solution. For other cycle times, EGA reached search spaces that SALOME cannot explore and find better solutions.

For the set of Hahn, the problem was solved with five different cycle times. EGA outperformed SALOME from SI perspective for all cycle times. Also, efficiency was improved at cycle times 2004 and 2806. Regarding other cycle times, EGA yielded same efficiency as SALOME.

This can be explained by the stochastic properties of Meta heuristics, which allow the search in different search spaces that cannot be reached by SALOME and find better SI and efficiency.

The sets of Tonge and Mukherje were solved using six different cycle times and seven cycle time, respectively. EGA outperformed SALOME at all cycle times from efficiency and SI perspectives.

SALOME starts building stations following dominance and reduction rules. Therefore, stations with maximum loads are built at the start of the line and stations with large idle times are built at the end of the line; whereas, EGA can explore different search spaces.

In order to find whether there is a significant difference or not between SALOME and EGA (for the five tested problem sets), paired t-test was conducted using R software as reported in Appendix A. we can see that, at 95% confidence level there is a significant difference between SALOME and EGA; efficiency and SI perspectives.

**Table-2: The solutions of Wee-Mag problem**

| Cycle time | SALOME | | | GA | | | RPW | | | EGA | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | n | E (%) | SI | n | E (%) | SI | n | E (%) | SI | n | E (%) | SI |
| 31 | 62 | 78 | 60.2 | 64 | 80.76 | 51.05 | 63 | 79.3 | 54.79 | 62 | 80.6 | 49.5 |
| 33 | 61 | 74.5 | 72.84 | 62 | 78 | 58.35 | 61 | 74.5 | 70.96 | 61 | 76.79 | 61.81 |
| 35 | 60 | 71.4 | 83.91 | 61 | 74.46 | 69.67 | 60 | 71.3 | 82.66 | 60 | 71.4 | 81.57 |
| 36 | 60 | 69.4 | 92.09 | 61 | 70.21 | 85.91 | 60 | 69.4 | 90.23 | 60 | 69.4 | 89.51 |
| 41 | 59 | 61.97 | 126.9 | 60 | 64 | 113.2 | 59 | 62 | 125.2 | 59 | 61.97 | 124.9 |

The genetic algorithm (GA) developed by [1] was coded using C++ for the reason of comparison with EGA.

Table-2 shows the solutions of Wee-mag problem with five different cycle times, using SALOME, GA, EGA and the heuristic method Ranked positional weight (RPW).

The heuristics in the initial population are the same in GA and EGA. Thus, RPW solution is already present in the population of two Meta heuristic algorithms. For 80% of these solutions, RPW reached the optimal solution; further, EGA reached the optimal solution of these five solutions. In contrast, GA reached inferior solutions to these found by SALOME, EGA and RPW. Additionally, GA outperformed other algorithms in SI and efficiency perspectives.

The rationale behind that is the large weight attributed to SI and small weight attributed to number of stations, in the objective function. Inferior solution are selected with smaller SI, which is found by increasing equity among stations and lowering cycle time. Besides, efficiency increases depending on minimizing the cycle time. Hence, lower SI and higher efficiency is not indicator for a lower number of work stations.

## CONCLUSIONS AND RECOMMENDATIONS

In order to highlight the assembly line balancing problem, EGA is proposed to solve single model problems with multi-objectives, which are the number of work stations and the smoothing index.

In order to show the efficacy of EGA, 30 optimal solutions (representing five problems) out of 171 solutions, were selected for comparison with solutions obtained from the state-of-the-art algorithm (SALOME), from efficiency and SI perspectives.

For small problems, at small cycle times, EGA becomes more restricted regarding cycle time and cannot find solutions with better efficiencies.

In view of the dominance and reduction rules like maximum load rule and Johnson rule, SALOME attempts to build Quasi fully-loaded stations at the start of the search and stations with high idle time at the end of the search; however, the stochastic properties of meta-heuristics can find solutions of better efficiencies and SI. Additionally, meta-heuristics can explore areas that SALOME cannot explore.

At 95% confidence level, concerning the 30 selected solutions, there is a significant difference between EGA and SALOME, from efficiency and SI perspectives.

Attributing large weight to SI and small weight to number of stations in objective functions can lead to select inferior solutions with higher number of work stations. Also, lower values of SI or higher efficiency is not an indicator of a lower number of work stations.

In summary, the enhanced algorithm showed efficacy in solving those problems and outperformed SALOME solutions from line efficiency and smoothing index perspectives.

Future research will focus on hybridizing exact and meta-heuristic methods, solving more complex problems with constraints and adding more objectives like minimizing cost or maximizing profit.

## Appendix A

```
        Paired t-test

data:  SALOME.eff and EGA.eff
t = -2.9785, df = 29, p-value = 0.005801
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -4.28242 -0.79558
sample estimates:
mean of the differences
                -2.539
```

**Fig-A1: t-test for efficiency between SALOME and EGA**

```
        Paired t-test

data:  SALOME.SI and EGA.SI
t = 2.4079, df = 29, p-value = 0.02263
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  20.72221 254.41645
sample estimates:
mean of the differences
              137.5693
```

**Fig-A2: t-test for SI between SALOME and EGA**

## REFERENCES


1. Leu, Y. Y., Matheson, L. A., & Rees, L. P. (1994). Assembly line balancing using genetic algorithms with heuristic-generated initial populations and multiple evaluation criteria. *Decision Sciences*, *25*(4), 581-605.
2. Rubinovitz, J., & Levitin, G. (1995). Genetic algorithm for assembly line balancing. *International Journal of Production Economics*, *41*(1-3), 343-354.
3. Kim, Y. K., Kim, Y. J., & Kim, Y. (1996). Genetic algorithms for assembly line balancing with various objectives. *Computers & Industrial Engineering*, *30*(3), 397-409.
4. Sabuncuoglu, I., Erel, E., & Tanyer, M. (2000). Assembly line balancing using genetic algorithms. *Journal of intelligent manufacturing*, *11*(3), 295-310.
5. Baykasoglu, A. (2006). Multi-rule multi-objective simulated annealing algorithm for straight and U type assembly line balancing problems. *Journal of Intelligent Manufacturing*, *17*(2), 217-232.
6. Cakir, B., Altiparmak, F., & Dengiz, B. (2011). Multi-objective optimization of a stochastic assembly line balancing: A hybrid simulated annealing algorithm. *Computers & Industrial Engineering*, *60*(3), 376-384.
7. Nearchou, A. C. (2011). Maximizing production rate and workload smoothing in assembly lines using particle swarm optimization. *International Journal of Production Economics*, *129*(2), 242-250.

8. Cheshmehgaz, H. R., Desa, M. I., & Kazemipour, F. (2012). A cellularrearranging of population in genetic algorithms to solve assemblyline balancing problem. *J Mech Eng Autom*, *2*, 25-35.
9. Scholl. (1999). Balancing and sequencing assembly lines, 2nd, Physica, Heidelberg.
10. Fathi, M., Fontes, D. B. M. M., Urenda Moris, M., & Ghobakhloo, M. (2018). Assembly line balancing problem: A comparative evaluation of heuristics and a computational assessment of objectives. *Journal of Modelling in Management*, *13*(2), 455-474.
11. Bierwirth, C., Mattfeld, D. C., & Kopfer, H. (1996, September). On permutation representations for scheduling problems. In *International Conference on Parallel Problem Solving from Nature* (pp. 310-318). Springer, Berlin, Heidelberg.
12. Coley, D. A. (1999). *An introduction to genetic algorithms for scientists and engineers*. World Scientific Publishing Company. Pte. Ltd.
13. Taha, R. B., El-Kharbotly, A. K., Sadek, Y. M., & Afia, N. H. (2011). A Genetic Algorithm for solving two-sided assembly line balancing problems. *Ain Shams Engineering Journal*, *2*(3-4), 227-240.