International Journal of Business and Management Invention (IJBMI)
ISSN (Online): 2319 – 8028, ISSN (Print): 2319 – 801X
www.ijbmi.org || Volume 7 Issue 4 Ver. II || April. 2018 || PP—28-35

Improving the Search Performance of Genetic Algorithm to Solve Assembly Line Balancing Problem

Mazouzi Mohamed[1],Belassiria Imad[1]
[1](*Mechanical department, ENSEM/ Hassan II University, Casablanca-Morocco)*
Corresponding Author: Mazouzi Mohamed

ABSTRACT: *Assembly line balancing belongs to a class of intensively studied combinatorial optimization problems known to be NP-hard in general. For the resolution of such problem, a genetic algorithm (GA) is implemented to solve type E of assembly line balancing problem (ALBP). The paper compares two kinds of GA, one is hybrid GA which hybridized traditional scheme of GA with heuristic priority rule-based procedure, and other one is GA adopted with strategy of localized evolution. Through computational experiments using ten test problems collected from literature, the search performance of traditional GA is improved and the comparisonresults between GAs is reported.*

Keywords: *Assembly line balancing problem; genetic algorithm; hybrid; strategy of localized evolution; performance comparison.*

Date of Submission: 04-04-2018 Date of acceptance: 19-04-2018

I. INTRODUCTION

An assembly line consists of a set of sequential workstations in which repetitive sets of tasks are performed in a limited time called cycle time. The classical problem of assigning tasks to the workstations, while one or more objectives are optimized without violating to respect main constraints and some specific restrictions imposed on the assembly line is called the assembly line balancing problem (ALBP).

In the literature, the ALBP is divided into four types of problems, these types are related to the objective function which evaluates the quality of feasible solutions. The basic optimization problems of ALPB are type I and type II, where the type I problem (ALBP-I) is the minimization of the number of workstations (*m*) for a given cycle time (*c*), and the type II problem (ALBP-II) is used to minimize the cycle time for a given number of workstations. While ALBP type E (ALBP-E) is implemented to maximize line efficiency (E) considering the interrelationship between *m* and *c*. Last kind of ALBP is a feasibility problem (ALBP-F) which is to establish whether or not a feasible line balance exists for a given combination of *m* and *c*.

Researchers have published many studies on the solution for the ALBP type I, e.g., (Bautista and Pereira [1], Özcan and Toklu [2], Essafi et al. [3], Kilincci [4]). However, the ALBP-I frequently occurs when the organization desires to design new assembly lines and the production demand can be easily estimated, therefore, the cycle time is pre-specified as a fixed input data. In the other words, the aim is to employ as few workstations (number of operators) as possible while meeting a requested level of production rate. Unlike ALBP type I, the goal of ALBP type II is to minimize the cycle time in order to maximize the level of production rate of an existing assembly line with a specific number of workstations. This type of problem may exist when there are some limitations in the assembly line which restricts the number of workstations, so that, the redesigns of an existing assembly line can be performed without utilizing new workstations, or when the organization tends to produce the maximum number of products by using existing resources without expansion. This type also has got a great attention in the literature, e.g., (Gao et al. [5], Kulak et al. [6], Kim et al. [7], Kilincci [8], Seyed-Alagheband et al. [9]).

The ALBP type E is more general problem which combines the ALBP type I and type II [10]. Therefore, the type E problem is presented when the production system is flexible enough to change the number of workstations and cycle time as well. However, it can be no possible to change those parameters arbitrarily. By demand rate, a maximum cycle time, which is the reciprocal of the production rate, is defined. Furthermore, the space available may set an upper limit on the number of workstations. Line efficiency can be defined with m and c being variables in certain ranges. This type of problem is less studied in literature because of its nonlinear form, such studies are treated in Wei and Chao [11], Scholl and Klein [12], Belassiria et al. [13]. In this paper, the objective function of the problem is a bi-objective function which aims to maximize line efficiency and balance workload between workstations simultaneously.

Due to the combinatorial nature of the ALBP which falls into NP-hard class of combinatorial optimization problems. The complexity to find an optimal solution increases when the instances include more

than a few tasks and/or workstations. Furthermore, in real-life situations one can observe numerous restrictions which make the ALBP more difficult to solve. Therefore, it can be time consuming for optimum seeking methods to obtain an optimal solution within this vast search space. However, many research efforts have been conducted towards the development of heuristics and meta-heuristics approaches such as simulated annealing (Suresh and Sahu [14]), tabu search (Peterson [15]) and genetic algorithms (Falkenauer and Delchambre [16]).

Genetic algorithm (GA) is one of meta-heuristics approaches and seems to be the most popular algorithm among them. GA received an increasing attention since it performs well for many types of optimization problems by using directed random searches to locale optimum solutions in complex problems. However, the standard scheme of GA may lack the ability of exploring the solution space effectively as problems get larger and more complex as in real life. Often, the pure GA is completed with some methods to improve its search ability. Over the last years, hybridization of GA has been considered as an important efficiency enhancement technique of GA [17]. In this paper, we present a hybrid GA (h-GA) in which the heuristic priority rule-based procedure is sequentially hybridized with GA, this kind of hybridization is used in Belassiria et al. [13]. Another approach that it can promote search efficiency is a localized evolution of genetic algorithm which is called a neighborhood GA (n-GA), this algorithm is adapted from Kim et al. [18]. This article compares the results obtained by h-GA and those obtained by n-GA at solving test problems which are widely used for ALBP.

The rest of this paper is organized as follow. The mathematical model is presented in section 2. Section 3 describes the two compared algorithms. Section 4 provides the experimental design and results. Conclusion is highlighted in section 5.

II. FORMULATION OF ALBP-E

The simpleALBP usually takes into account two constraints (the precedence constraints and the cycle time). In industry there are more realistic situations which make the ALBP hardly applicable. Therefore, we tackle here some of additional constraints, which commonly has been used in real practices, in order to help line managers to deal with assignment difficulties.

The notations and mathematical model used in this paper for the two algorithms are adapted from Belassiria et al. [13].

1. Notation

I: set of tasks; $I = \{1, 2, \ldots, i, \ldots, n\}$.
J: set of workstation; $J = \{1, 2, \ldots, j, \ldots, m\}$.
C: cycle time.
t_i: processing time for task *i*.
F_i: set of successors of task *i*.
P_0: set of tasks that has no immediate predecessors; $P_0 = \{i \in I | P(i) = \emptyset\}$.
W_j: subset of all tasks that can be assigned in workstation *j*.
$\|W_j\|$: number of tasks that can be assigned in workstation *j*.

2. Mathematical model

Decision variables:

$$x_{ij} = \begin{cases} 1 & \text{if task i is assigned to workstation j} \\ 0 & \text{otherwise.} \end{cases}$$

$$U_{ij} = \begin{cases} 1 & \text{if task i can be assigned to workstation j} \\ 0 & \text{otherwise.} \end{cases}$$

The ALBP-E model combines the SALBP-I and SALBP-II which is defined as $= \frac{t_{sum}}{m \times C}$, so the idle time is $(m \times C) - t_{sum}$ .The objective function of the problem is to maximize assembly line efficiency and simultaneously minimize the idle time, and furthermore it can be achieved by minimizing the number of workstations and cycle time.

$$WE = \frac{\sum_{i=1}^{n} t_i}{m \times C};\ i \in I \qquad (1)$$

The assembly line efficiency is always in the value of (0, 1) and it is a maximization function. When the efficiency closer to 1, the assembly line will has better performance in which it has lower idle time.

$$Max\ F = WE \qquad (2)$$

In addition to the cycle time and precedence constraints, we have considered additional constraints, in which we can find in realistic situations. So the constraints of the mathematical model are as follows:

Subject to

$$\sum_{j=1}^{m} x_{ij} = 1, \quad i \in I \qquad (3)$$

$$\sum_{j=1}^{m} x_{bj} - \sum_{j=1}^{m} x_{aj} \leq 0 \quad a \in I, b \in F_a \qquad (4)$$

$$\sum_{i=1}^{n}(t_i x_{ij}) + s_j \le C \quad (5)$$
$$\sum_{j=1}^{m} x_{aj} - \sum_{j=1}^{m} x_{bj} = 0, \ (a,b) \in ZP \quad (6)$$
$$x_{aj} - x_{bj} \le 1, \ (a,b) \in ZN, j = 1,\dots,m \quad (7)$$
$$\sum_{i \in I} x_{ij} - \|W_j\| U_{ij} \le 0 \quad (8)$$
$$x_{ij} \in \{0,1\}, \forall\, i \in I, \forall\, j \in J \quad (9a)$$
$$s_j \ge 0, \forall\, j \in J \quad (9b)$$

Constraint (3) ensures that each task is assigned to only one workstation. Constraint (4) describes that the assignment of a task must be follow the precedence constraint, i.e. a task can only be assigned when its entire predecessors are finished. Constraint (5) ensures that the sum of processing time and idle time in a workstation must be smaller than or equal to the cycle time. Constraint (6) is compatibility zoning constraint, this type of constraints are normally related with the use of common equipment or tooling. For example, if two tasks need the same equipment or have similar processing conditions it is desirable to assign these tasks to the same workstation. Constraint (7) is incompatibility zoning constraint. These constraints are usually imposed when a set of tasks require different equipment, thus they cannot share the same workstation. Or when it is not possible to perform some tasks in the same workstation for safety reasons. Constraint (8) ensures that specific tasks need to be assigned into specific workstation. This is usually due to some kind of heavy equipment that would be too expensive to move elsewhere in the shop.Equations (9a)-(9b) define domain of the decision variables.

III. GENETIC ALGORITHM FOR SOLVING ALBP-E

As mentioned above, GAs have been proven to be very efficient and powerful in a wide variety of application particularly in combinatorial optimization problems. Another advantage is that GA has the strength that it is flexible in dealing with various types of optimization criteria and constraints.

1. Hybrid genetic algorithm (h-GA)

Since the ALB problem in this study considers a large number of tasks and precedence restrictions. The solution of the algorithm should seek to assign tasks to workstations in order to find the optimal objective function, the algorithm's efficiency is highly dependent on the initial solution. Therefore, by using a heuristic priority rule-based procedure, mixed with generating random solutions a more rich initial seeds is created. The task assignment rules used in our h-GA is proposed by Baykasoglu and Özbakir [20], and a list of task assignment rules is shown in Table 1.

The flow chart of the proposed h-GA is presented in Fig. 1. The initialization of h-GA contains the creation of the initial individuals by heuristic priority rule-based procedure and random individuals. Besides, genetic operators are used to improve the solution quality.

The proposed hGA is outlined step by step as follows:

Step 1: Generate an initial population of individuals obtained by the task assignment rules.
Step 2: Decode all individuals and evaluate the fitness function of their corresponding solutions.
Step 3: Determine pairs which define which chromosomes will undergo crossover (selection)
Step 4: Apply crossover operator to the selected pairs in order to obtain new pairs of chromosomes (offsprings).
Step 5: Apply mutation operator to one parent
Step 6: Decode all offspring and evaluate the fitness function of their corresponding solutions
Step 7: Decide which chromosome will form new generation; to preserve the diversity of the population, the offspring, which has not duplicated with any solution in the population, replace with the poor solution.
Step 8: If termination criterion is satisfied go to Step 9, else go to Step 3.
Step 9: Terminate the algorithm and present best solution.

2. Neighborhood genetic algorithm (n-GA)

Flow diagram of the proposed n-GA is depicted in Fig. 2. Initialisation of the n-GA contains generating the initial population randomly. Afterword, we adopt the strategy of localized evolution, presented by (Kim et al. [18]), that we expect can promote population diversity and search efficiency. The population forms a two dimensional structure of toroidal grids. The structure of a 3×3 neighborhood is used here for the localized evolution. Let NHst denote the neighborhood including individual (s, t) and its eight neighbors in the population. The evolution of NHst proceeds while interacting with individuals within NHst.

The procedure of the n-GA is outlined step as follows:

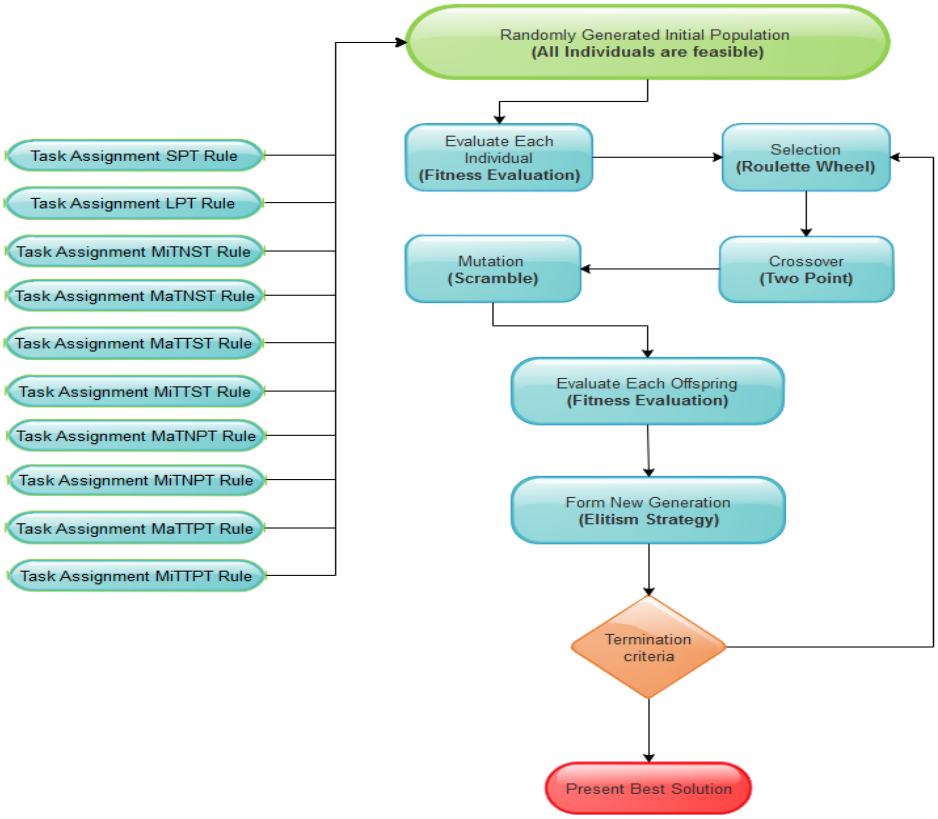

Fig. 1. Flow diagram of the proposed hybrid genetic algorithm

TABLE I. LIST OF TASK ASSIGNMENT RULES

Rule no.	**Task assignment rules**
1	Shortest Processing Time (SPT)
2	Longest Processing time (LPT)
3	Minimum Total Number of Successor Tasks (MiTNST)
4	Maximum Total Number of Successor Tasks (MaTNST)
5	Minimum Total Time of Successor Tasks (MiTTST)
6	Maximum Total Time of Successor Tasks (MaTNST)
7	Minimum Total Number of Predecessor Tasks (MiTNPT)
8	Maximum Total Number of Predecessor Tasks (MaTNPT)
9	Minimum Total Time of Predecessor Tasks (MiTTPT)
10	Maximum Total Time of Predecessor Tasks (MaTTPT)

Step 1: Generate a random initial population of population.
Step 2: Decode all individuals and evaluate the fitness function for each individual. Set fbest to the best individual.
Step 3: Divide the population into numerous cell.
Step 4: An arbitrary cell (s, t) is selected, which sets up NHst.
Step 5: Evolution of NHst
Step 5.1: The fitness of each individual in NHst is evaluated.
Step 5.2: Apply crossover operator to the selected parents in order to produce two offsprings.
Step 5.3: Two individuals which have the worst fitness in NHst are replaced with the offspring newly produced.
Step 4: An arbitrary cell (s, t) is selected, which sets up NHst.
Step 5: Evolution of NHst
Step 5.1: The fitness of each individual in NHst is evaluated.
Step 5.2: Apply crossover operator to the selected parents in order to produce two offsprings.
Step 5.3: Two individuals which have the worst fitness in NHst are replaced with the offspring newly produced.
Step 5.4: Apply mutation operator to the individuals in NHst.

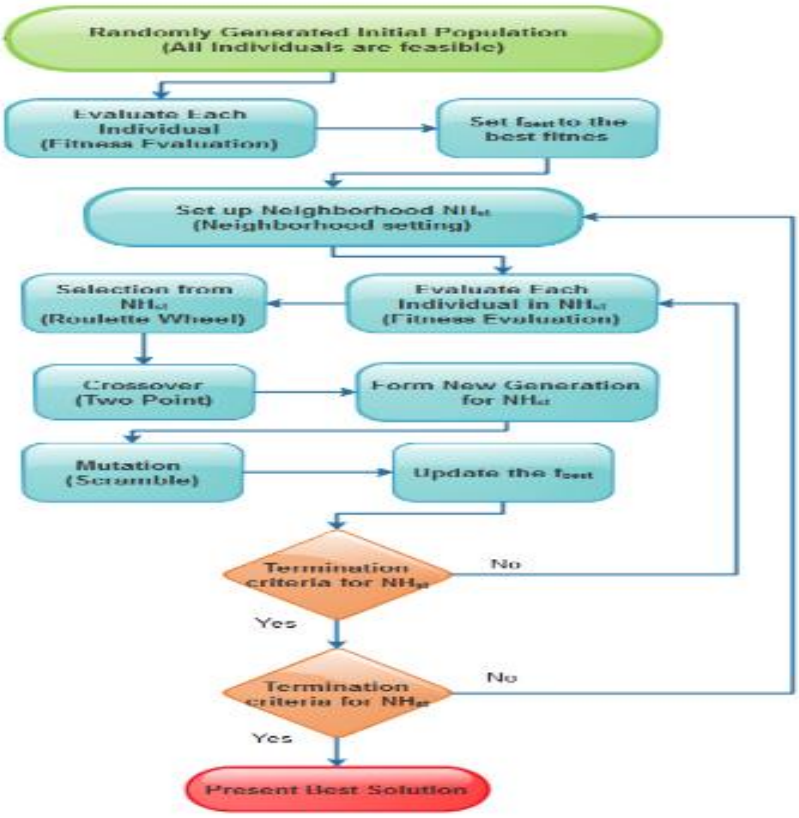

Fig. 2. Flow diagram of neighborhood genetic algorithm

Step 5.5: The fitness function is computed for the newly produced individuals using the decoding method. If the best fitness in the newly evaluated individuals bigger than fbest, then fbest is updated with the new best fitness.
Step 5.6: If the termination criteria for NHst is met, then go to Step 6. Otherwise, go to Step 5.1.
Step 6: If the termination criteria is satisfied, then stop. Otherwise, go to Step 4.

IV. REPRESENTATION AND INITIALISATION

For solving the ALBP-E by h-GA and n-GA, a well-known task based representation is employed to represent chromosome structure (Sabuncuoglu et al. [19]). An individual is a string of length n (the number of tasks), and each gene of the chromosome represents a task. Therefore, tasks assignment to the workstations is according to the task sequence in the chromosome. For example, Fig.3. illustrates assignment of tasks to workstations according to a chromosome for a small-sized problem consists of 13 tasks.

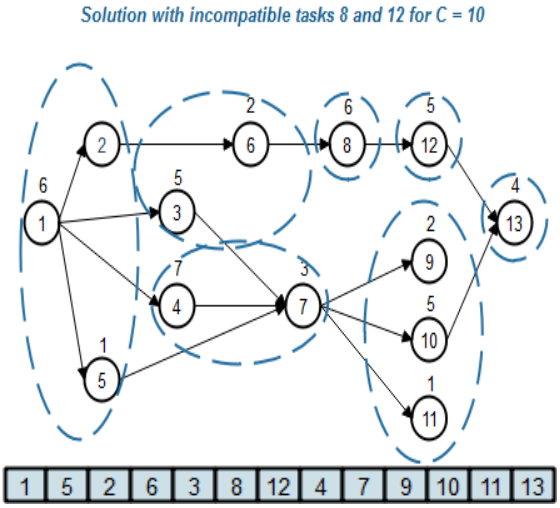

Fig. 3. Assignment procedure according to chromosome

To generate individuals in the initial population task assignment heuristics method, the following procedure is iterated.
Step 1: All tasks that have not been assigned are idendified. Find a set of tasks, F, such that all their predecessors have been assigned..
Step 2: Selecting a task *i* in F based on the selected task assignment rule. If there are more than one task with the same priority, a task is selected randomly.
Step 3: checking all the task constraints.
Step 4: If all constraints are fulfilled, task *i* is assigned to the workstation and then go to step 5. Otherwise, remove the task from F, and then go to step 2.
Step 5: If all tasks have been assigned, then stop. Otherwise, go to step 1.

3. Genetic operators

a. Crossover operator

We use a special two-point crossover method adapted from Belassiria et al. [13] as shown in Fig 4. in order to generate offspring from parents, in which these two points are randomly generated.
The procedure of the crossover operation is described as follows.
Step 1: One integer value r between [1, *n*-1] are generated randomly.
Step 2: The genes with values of [1, *r*] are copied into the same position in child.
Step 3: The alteration procedure is used to fill the second part in Offspring 1.
Step 3.1: Identifying all tasks that have not been assigned in the child, UT.
Step 3.2: Starting right from the cut point of the copied part.
Step 3.3: Select task *i* in UT, using the order of the second parent.
Step 3.4:Wrapping around at the end.
Step 4: Analogous for the second child, with parent roles reversed.

b. Mutation operator

Here we use scramble mutation adapted from Belassiria et al. [13]. One number between 1 and the chromosome length is arbitrarily generated. Chromosome parent is separated into two sections: head and tail. The new mutated offspring keeps the head part of the parent, and the tail part is filled using the same procedure used to generate initial population with random assignment. Fig 5. represents a scramble mutation method.

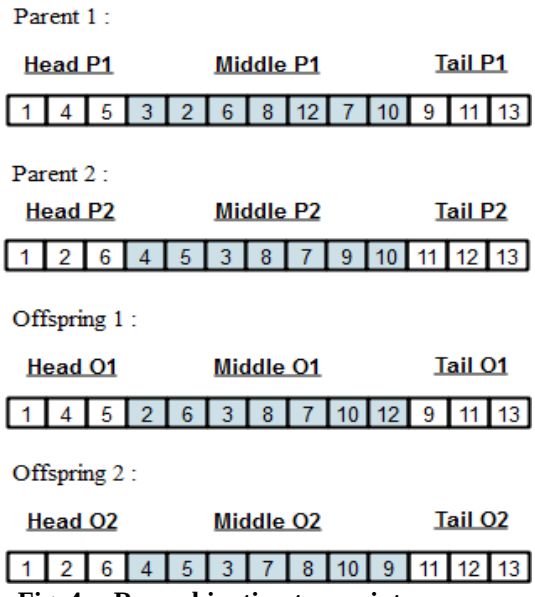

Fig. 4. Recombination two point crossover

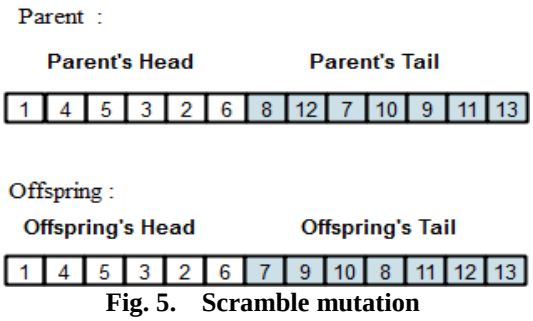

Fig. 5. Scramble mutation

V. THE EXPERIMENTS AND RESULTS

In order to assess the efficiency of the compared algorithms, it is tested on the data set taken from Scholl [21]. This problem set involves ten test problems and the summaryof the data set is given in Table 2. For each problem, we give the number of tasks *N*, the cycle time *C* and the minimal number of workstation *m**. If *m** is not equal to the theoretical minimum number of workstation m_{min}: = $[t_{sum}/C]$, the symbol '+' is added. All experiments are executed on a personal computer with an EliteBook HP with 2.50 GHz i5-2520M CPU. The algorithms were coded by C# 6.0 (visual studio 2013). We set a genetic parameters used for h-GA and n-GA as follows: crossover rate = 0.5, mutation rate = 0.15, and the population size P_z = 100. The number of necessary iterations is varied related to size problem.

TABLE II. COMPUTATIONAL RESULTS FOR THE TEST PROBLEMS

	Problem No	*N*	*C*	Graph	*m**	Pure GA		h-GA		n-GA	
						m	*WE(%)*	*m*	*WE(%)*	*m*	*WE(%)*
Medium-size	1	28	216	Heskia	5	5	88,148	5	89,259	5	90,000
	2	32	2020	Lutz1	8+	8	87,500	8	87,500	8	87,500
	3	35	54	Gunter	9	11	81,313	10	83,333	10	83,333
Large-size	4	53	2806	Hahn	6	6	83,310	6	83,310	6	83,310
	5	58	92	Warnecke	17	20	84,130	20	84,130	20	84,130
	6	70	320	Tonge	11	13	84,375	12	91,406	12	91,406
	7	89	20	Lutz2	25	27	89,815	27	89,815	27	89,815
	8	94	281	Mukherje	16+	17	88,089	16	93,594	16	93,594
	9	148	564	Bartholdi	10	12	83,244	11	90,490	12	83,244
	10	297	1515	Scholl	46-47	52	88,623	51	90,360	52	88,623

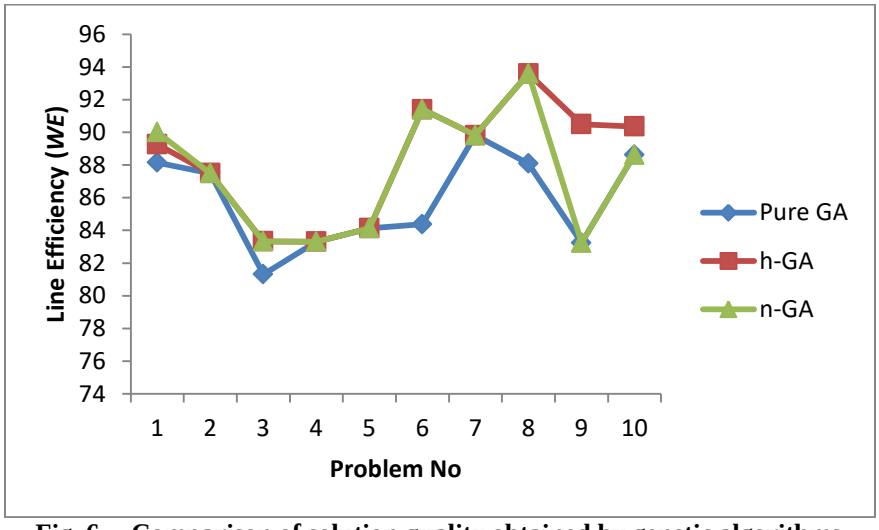

Fig. 6. Comparison of solution quality obtained by genetic algorithms

Table 2 exhibits results obtained for the 10 problems by the pure GA, h-GA and n-GA in terms of the line efficiency (*WE%*). To maximize line efficiency one of the parameters *m* or *c* is to modify. In this case, the cycle time is given and the number *m* of workstation is minimized.

The experiments for the compared GAs are repeated 10 times and the best solution is taken. In the experimental results show that for the most of experimental problem, the two GAs (h-GA and n-GA) provide better outcomes than pure GA, which conforms that the hybridization technique and local search strategy applied to GA maintain the global diversity and improve search performance of pure GA to find a better performance specially in more complex problems. On the other hand, if the line efficiency obtained by h-GA and n-GA for problems 1, 2, 3, 4, 5, 6, 7 and 8 are compared, the h-GA outperforms the n-GA for the large problems 9 and 10. Therefore, Sequential hybridization can provide more performance for large size problem.
To analysis the effect of hybridization technique and strategy of localized evolution adapted to GA on the genetic search. Fig 6. shows the performance comparison of the compared GA techniques. The curves in Figure 6 correspond to the line efficiency of the best solution attained by the three GAs over each one of the test problems. As can be seen from this figure, seeding the initial population with good solution using a mixed quantity of solutions obtained by the assignment rules heuristics and random assignment improves significantly the search performance of h-GA resulting to higher quality solutions for the large problems.

VI. CONCLUSION

This paper addressed ALBP type E with the objective of maximizing line efficiency. Through computational experiments, we presented a comparison of two famous technique applied to genetic algorithm to improve search performance: hybridization and localized evolutionary. Although the h-GA and n-GA showed a better performance than traditional GA for all test problems, which conforms the effectiveness of these techniques to improve the solution quality of traditional GA. Compared between the two techniques, it's observed that h-GA increase the solution quality especially for large-size problems.

REFERENCES

[1] Bautista, J., Pereira, J., 2009. A dynamic programming based heuristic for the assembly line balancing problem. European Journal of Operational Research 194 (3), 787–794.
[2] Özcan, U., & Toklu, B. (2009). Balancing of mixed-model two-sided assembly lines.Computers & Industrial Engineering, 57, 217–227.
[3] Essafi, M., Delorme, X., Dolgui, A., & Guschinskaya, O. (2010). A MIP approach for balancing transfer line with complex industrial constraints. Computers &Industrial Engineering, 58, 393–400.
[4] Kilincci, O., 2011. Firing sequences backward algorithm for simple assembly line balancing problem of type 1. Computers & Industrial Engineering 60 (4),830–839.
[5] Gao, J., Sun, L., Wang, L., Gen, M., 2009. An efficient approach for type II robotic assembly line balancing problems. Computers & Industrial Engineering 56 (3), 1065–1080.
[6] Kulak, O., Yilmaz, I., Gunther, H., 2008. A GA-based solution approach for balancing printed circuit board assembly lines. OR Spectrum 30 (3), 469–491.
[7] Kim, Y., Song, W., Kim, J., 2009. A mathematical model and a genetic algorithm for two-sided assembly line balancing. Computers and Operations Research 36 (3), 853–865.
[8] Kilincci, O., 2010. A Petri net-based heuristic for simple assembly line balancing problem of type 2. International Journal of Advanced Manufacturing Technology 46 (1), 329–338.
[9] Seyed-Alagheband, S., Fatemi Ghomi, S., Zandieh, M., 2011. A simulated annealing algorithm for balancing the assembly line type II problem with sequencedependent setup times between tasks. International Journal of Production Research 49 (3), 805–825.
[10] Scholl, A., 1999. Balancing and sequencing assembly lines. 2nd ed. Heidelberg, Germany: Physica.
[11] Wei, N.C., Chao, I.M., 2011. A solution procedure for type E simple assembly line balancing problem. Computers & Industrial Engineering 61 3, 824–830.
[12] Scholl, A., Klein, R., 1999. ULINO: optimally balancing U-shaped JIT assembly lines. International Journal of Production Research 37, 721–736.
[13] Belassiria, I., Mazouzi, M., El Fezazi, S., & El Maskaoui, Z. (2017, April). An efficient approach for workload balancing of assembly line systems with assignment restrictions. In Logistics and Supply Chain Management (LOGISTIQUA), 2017 International Colloquium on (pp. 7-12). IEEE.
[14] Suresh, G., & Sahu, S. (1994). Stochastic assembly line balancing using simulated annealing. International Journal of Production Research, 32(8), 1801–1810.
[15] Peterson, C. (1993). A tabu search procedure for the simple assembly line balancing problem. In The proceedings of the decision science institute conference (pp. 1502–1504). Washington, DC.
[16] Falkenauer, E., & Delchambre, A. (1992). A genetic algorithm for bin packing and line balancing. In The proceedings of the 1992 IEEE international conference on robotics and automation (pp. 1189– 1192). Nice, France.
[17] Akpınar, S., Bayhan, G.M., 2011. A hybrid genetic algorithm for mixed model assembly line balancing problem with parallel workstations and zoning constraints. Engineering Applications of Artificial Intelligence 24 (3), 449–457.
[18] Kim, Y., Song, W., Kim, J., 2009. A mathematical model and a genetic algorithm for two-sided assembly line balancing. Computers and Operations Research 36 (3), 853–865.
[19] Sabuncuoglu, I., Erel, E. & Tanyer, M. Assembly line balancing using genetic algorithms. Journal of Intelligent Manufacturing (2000) 11: 295
[20] Baykasoglu A, Özbakir L (2007) Stochastic U-line balancing using genetic algorithms. Int J Adv Manuf Technol 32(1–2):139–147.
[21] A. Scholl, "Data of assembly line balancing problems," (working paper), TH Darmstadt, 1993. Available: http://www.assembly-line-balancing.de/

Mazouzi Mohamed, and Belassiria Imad." Improving the Search Performance of Genetic Algorithm to Solve Assembly Line Balancing Problem " International Journal of Business and Management Invention (IJBMI) , vol. 07, no. 04, 2018, pp. 28–35.