International Journal of Industrial Engineering & Production Research

March 2015, Volume 26, Number 2
pp. 105-117

http://IJIEPR.iust.ac.ir/

pISSN: 2008-4889

A Bi-Objective Approach for Design of an Assembly Line Re-Balancing System: Mathematical Model and Differential Evolution Algorithms

Hadi Mokhtari , Ashkan Mozdgir

Hadi Mokhtari *Assistant Professor, Department of Industrial Engineering, Faculty of Engineering, University of Kashan, Kashan, , Iran*
Ashkan Mozdgir *PhD Candidate, Faculty of Industrial Engineering, K.N. Toosi University of Technology, Tehran, Iran.*

KEYWORDS

Differential Evolution,
Assembly Lines,
Scheduling of Work Station,
Taguchi Method,

ABSTRACT

In many practical systems, configuration of assembly lines is fixed and designing a new line may be incurred huge amount of costs, and thereby it is not desirable for practitioners. Hence, in this paper we suggest a re-balancing optimization model of an existing assembly line in which a new demand related cycle time (CT) is needed. The objective is to re-schedule the tasks in order to reduce the current CT to the new required one such that two objectives are optimized: (i) minimization of the incurred costs and (ii) minimization of non-smoothing of reconfigured line. To solve the considered problem, a differential evolution algorithm (DEA), as an effective meta-heuristic, is adopted. The algorithm is easy to implement and requires a few control parameters. It generally shows reliable and accurate performance. However, the user is required to set the values of control parameters for each problem which is a time consuming procedure. To overcome this drawback and enhance the performance of DEA, its parameters are optimized by the use of Taguchi method which is an efficient statistical technique for parameter design. The obtained results from computational experiments on benchmark instances show the effectiveness of suggested algorithm against other methods..

1. Introduction

Assembly lines are special kind of production flow-lines extensively used in mass production systems to increase efficiency and speed, decrease per-unit cost, and make ease to monitor and control manufacturing process. Usually an assembly line consists of a set of sequential workstations that are typically connected by a continuous material-handling system. Among the decision making problems usually occurred in such systems, balancing of assembly line is one of important task dealt with by both researchers and practitioners.

* **Corresponding author: Hadi Mokhtari**
Email: mokhtari_ie@yahoo.com

This problem is called the assembly line balancing problem (ALBP) and usually occurred when an assembly line has to be configured or redesigned. The aim of this problem is to arrange the individual processing and assembly tasks (jobs) at the workstations so that one or more specific objective(s) is satisfied. According to the assumptions, ALBP can be classified into two main categories: (i) simple ALBP (SALBP), and (ii) generalized ALBP (GALBP). Four types of formulation have been yet presented with SALBP: (i) SALBP-1 which attempts to minimize the number of stations for a given fixed cycle time, (ii) SALBP-2 which attempts to minimize the cycle time of the line for a given number of stations, (iii) SALBP-E which attempts to maximize the line efficiency upon a feasible solution and SALBP-F which consists of determining if a certain assembly solution is feasible or not. Moreover the GALBP is an extended type of ALBP with different additional characteristics such as cost functions, equipment selection, paralleling, U-shaped line layout, and mixed-model production. The first research published on the assembly line balancing problem is Salveson [1] in 1955. After that the ALBP has been a hot topic for researchers. Different optimal solution techniques, such as branch-and-bound procedures [14,15] and dynamic programming [16, 17] have been developed for assembly line balancing problems. A good review of these techniques can be found in Scholl and Becker [18]. Regarding the solving methods, due to the combinatorial nature of the problem, applicability of optimal solution algorithms for large-sized problems is restricted. Even the linear assembly line balancing is known to be a NP-hard problem [19]. If there are m tasks and r ordering constraints, then there are $m!/2^r$ possible task sequences [20]. Because of such a vast search space, it is nearly impossible to obtain an optimal solution by deterministic algorithms. Therefore, recently, some researchers turned their attention to the use of meta-heuristics for the solution of SALBP. The most notable of this group of algorithms are evolutionary computation (EC) methods such as evolutionary algorithms [21], simulated annealing [22] and tabu-search [23, 24]. Nearchou [25] uses differential evolution in order to solve multi-objective assembly line. His Extended comparisons with other previously published evolutionary computation methods showed a superior performance for the proposed approach.

Despite the great interest in ALBP demonstrated by researchers [2-5], recently Ghosh and Gagnon [6] and Erel and Sarin [7] talked about the inability of the published methods to correctly model actual conditions and proposed that further works should be conducted at more useful directions in which the impact of real-life variables on assembly lines is considered. It was shown that in spite of the great amount of extensions of basic assembly line balancing, there remains a gap between requirements of real configuration problems and the status of research papers. Rekiek (2000) [8] mentioned that the gap between research methodologies developed for ALBP and real-life problems is caused by a number of factors. It might result from research papers focusing on just a single or only a few real extensions at a time. As an example, in a modern production system, not only ALBPs occur prior to its construction, but also due to dynamic nature of market parameters, a reconfiguration or rebalancing is needed continuously during the system lifetime [9, 10]. In literature, a wide variety of algorithms for solving the traditional ALBP are found, however almost all of them consider this problem under static condition that is, before the line deployment [11]. Nevertheless, need for improvement in design of product, continuous changes in market requirements, presenting more options for customer and also tendency to reduce time-to-market are some factors encourage the researcher for a dynamic version of ALBP. Since unpredictable changes are inevitable, in order to increase capability of responding to abovementioned changes, this problem are becoming more desirable in recent years. In order to formulate the suggested ALBP, its objective and constraints should be delineated. During the lifetime of a production system, when a market parameter is changed, the minimization of the number of stations is less important because of its high expenses. On the other hand, the cycle time is often determined based on sales forecasts. As

a consequence, the major decision in an ALRBP is to find a feasible solution in which the given number of stations and the demand related cycle time are not violated. Equivalently the number of workstations and required cycle time are considered as new constraints in modeling of an ALRBP in addition to the precedence constraints and assignment constraints (an assignment of each task to exactly one station). Once resources are allotted to stations, heavy machinery might not be reallocated. In this case, all tasks which require this resource need to remain at their previous station, which can be enforced by assignment restrictions [3]. Additionally, the movement of a machine is, however, not technically impossible, but associated movement costs are inevitable. In this case, movement costs need to be considered explicitly [11, 12]. Moreover, space constraints need to be considered whenever a machine is moved [13]. The space at a station might be limited, so that two tasks each of which requires a large machine cannot be assigned to the same station [3]. Not only the machinery, but also the operators of the assembly line have been assigned to a certain station. They may have been especially trained to carry out the respective work content, so that a change will additionally be associated with training costs. Therefore it could be desirable that the new line balance remains as close as possible to the previous one, in order to save training and movement costs. Hence the objective of suggested ALRBP is maximized the similarity between current line and the new one in terms of assigned tasks to the stations. Therefore, we can infer that the suggested ALRBP is like ALBP except that number of station is fixed and a new demand related cycle time should be satisfied. It does not focus on balancing a new line; instead it considers a more realistic problem which is re-balancing an existing line. ALRBP can be classified to simple and general ALRBP which consider different additional characteristics, as ALBP has been classified. In this paper, differential a evolution algorithm is developed to solve simple assembly line re-balancing problem (SALRBP). Furthermore, to enhance the performance of algorithm, its parameters are optimized using Taguchi method which is a conventional statistical technique for parameter design.

The rest of paper is structured as follow: At first, the problem is described in next section. Differential evolution is explained in section 3. Taguchi experimental design which analyzed suggestive algorithm, present in section 4; meanwhile, summery and possible future research directions are provided in Section 5.

2. Problem Definition and Mathematical Modeling

SAL re-balancing can be stated as follows. A set $WS = \{1,\dots,m\}$ of m workstations are arranged along an assembly line. There are n elementary operations called tasks $V = \{1,\dots n\}$ should be carried out by workers. Each task i $(i \in V)$ is performed on exactly one workstation and requires a deterministic processing time t_i $(i = 1,\dots,n)$. The tasks are partially ordered by precedence relations defining a directed acyclic precedence graph (DAG) $G = (V, E)$. An edge (i, j) denotes that task i must be started before task j can be started. Each station k has a station load S_k (i.e. a set of tasks assigned to station k). Each workstation can complete its assigned tasks within the specified cycle time. $tS_k = \sum_{i \in Sk} t_i (i = 1,2,...,m) < CT$. The cycle time which denoted by c is determined based on the sailing factors. There is an initial assignment in order to assign tasks into workstations and an initial cycle time which denotes by $\hat{c}$. Given WS, t_i $(i = 1,\dots,n)$ and G , the objective of SAL re-balancing problem is to find a feasible line balance (i.e., an assignment of the n tasks to the m workstations not violating the precedence and assignment constraints) in which the cycle time is less than c and remains as similar as previous one. The following assumptions are used in the formulation of the model.

(1) The assembly line contains one family of products.
(2) All the tasks are performed through a predefined manner.
(3) A new cycle time related to the market demand is needed which is different from current cycle time.

(4) The assembly line has a straight shape and there is not any buffer space.
(5) All the tasks should be performed based on precedence constraints.
(6) All the processing times are deterministic.
(7) All the machines are available at planning horizon.

According to above statement, mathematical model of the SALRBP can be proposed as follows:
Let x_{ik} is a binary variable such that:

$$x_{ik} = \begin{cases} 1 & \text{if task } i \text{ is a} \\ 0 & \text{otherwise} \end{cases} \quad \text{for } i = 1,2,\dots,n \text{ and } k \in W$$

$$\sum_{k=1}^{m} x_{ik} = 1 \qquad \text{for } i = 1,2,\dots,n \qquad (1)$$

$$\sum_{i=1}^{n} t_i x_{ik} \le c \qquad \text{for } k = 1,2,\dots,m \qquad (2)$$

$$\sum_{k \in WS_h} k x_{hk} \le \sum_{k \in WS_i} k x_{ik} \quad \forall (i,h) \in A \,\&\, L_h \ge E_i \qquad (3)$$

where:
M Precedence matrix
G Precedence graph
V Set of all tasks
n Number of tasks
m Number of stations
CT The cycle time which is determined based on the sailing factors
WS_i The station interval of the task i

$$WS_i = [E_i, L_i] \qquad (4)$$

E_i The earliest station of the task i, for i= 1,2, ... , N

$$E_i = \left[(t_i + \textstyle\sum_{h \in P_i} t_h)/CT\right] \qquad (5)$$

P_i Set of direct and indirect predecessors of task i, with respect to M
L_i The latest station of the task i, for i= 1,2, ... , N

$$L_i = m + 1 - \left[(t_i + \textstyle\sum_{h \in F_i} t_h)/CT\right] \qquad (6)$$

F_i Set of direct and indirect successors of task i, with respect to M
A Set of direct precedence relations $= \{(i,j) | i \in V \text{ and } j \in F_i\}$

The constraints (1) ensures that each task is assigned to exactly one station, constraint (2) guarantees that the new cycle time is not exceeded and constraint (3) represents restrictions due to technological precedence among tasks. Feasible solutions are evaluated latter with the following objective:

$$\max \quad NSS = \textstyle\sum_{i=1}^{n} y_i \qquad (7)$$

$$s.t.$$

$$y_i = \textstyle\sum_{k=1}^{m} x_{ik} x'_{ik} \quad \text{for } i = 1,2,\dots,n \qquad (8)$$

where $\acute{x}_{ik}$ is a binary variable so that:

$$x'_{ik} = \begin{cases} 1 & \text{if task } i \text{ is assigned to station } k \text{ in initial assignment} \\ 0 & \text{otherwise} \end{cases}$$

and NSS is number of tasks which have been assigned to their previous stations.

3. Solution Approach

3-1.General Differential Evolution Algorithm (DEA)

The differential evolution algorithm (DEA) is a kind of evolutionary algorithms (EA) first introduced by Storn and Price [26]. Due to its invention, DEA has been extensively applied with high success on many numerical optimization problems outperforming other more popular population heuristics such as GAs [27, 28]. Recently, some researchers successfully extended the application of DEA to the machine layout problem [29] and the flow-shop scheduling problem [30].
The main feature and stages of a classical DEA are as follows:

Step 1: DEA utilizes Np, D-dimensional parameter vectors $x_{i,k}$, i=1, 2, . . . , Np, as a population to search the feasible region Ω uniformly of a given problem. The index k denotes the iteration (or generation) number of the algorithm. The initial population (where k=0),

$$\varphi = \{x_{1,0}, x_{2,0}, \dots, x_{NP,0}\} \qquad (9)$$

Step 2: At each iteration all vectors in φ are targeted for replacement. Therefore, Np

competitions are held to determine the members of φ for the next iterations. This is achieved by using mutation, crossover and acceptance operators.

Step 3: Mutation phase
For each target vector $x_{i,k}$, i=1, . . . , Np, a mutant vector $\hat{x}_{ik}$ is obtained by:

$$\hat{x}_{i,v} = r.x_{best,k} + (1-r)\left(x_{\alpha,k} + Fs\left(x_{\beta,k} - x_{\gamma,k}\right)\right) \qquad (10)$$

where $\alpha, \beta, \gamma \in \{1, \ldots, Np\}$ are mutually distinct random indices and are also different from the current target index i, $r \in [0,1]$ is a random number that is generated randomly for each mutant vector. It is a coefficient for the convex combination between the best element $x_{best,k}$ of φ, and a randomly combination of three random elements. The vector $x_{\alpha,k}$ is known as the base vector and Fs>0 is a scaling parameter.

Step 4: Crossover phase
The crossover operator is applied to obtain the trial vector $y_{i,k}$ from $\hat{x}_{ik}$and $x_{i,k}$ by the following equation:

$$y_{i,k}^{L} = \begin{cases} \hat{x}_{i,k}^{L} & if\ R^{L} \le CR \text{ or } L = I_i \\ x_{i,k}^{L} & otherwise \end{cases} \qquad (11)$$

In the above equation CR shows the crossover rate, I_i is a random number in [1,D] that guarantee that at least one gene is difference between $x_{i,k}$ and $y_{i,k}$ and therefore $x_{i,k}$ and $y_{i,k}$ are not the same no way.

Step 5: Acceptance phase
After all Np trial vectors $y_{i,k}$ have been generated, acceptance is applied. In the acceptance phase, the fitness function of the trial vector, $F(y_{i,k})$, is compared to $F(x_{i,k})$, the value at the target vector and the target vector is updated using:

$$x_{i,k+1} = \begin{cases} y_{i,k} & if\ F(y_{i,k}) < F(x_{i,k}) \\ x_{i,k} & O.W \end{cases} \qquad (12)$$

Step 6: Mutation, crossover and acceptance phases continue until some stopping conditions are met.

3-2. The Proposed DEA-Based Approach for the Solution of SAL Re-balancing Problem

Adapting a DEA for a particular domain needs the specification of the following characteristics:
– A representation mechanism, i.e., a way of encoding ALB solutions to floating-point vectors.
– An evaluation mechanism, i.e., a way of evaluating the quality of each vector.
– A way of initializing the population of vectors.
– The application of mutation and crossover operators on the population in order to generate new 'better' populations.

3-3. The Representation Mechanism

The representation mechanism of SAL re-balancing problem is similar to SALB. Two different schemes of string representations applicable to ALBPs: The station-oriented and the task-oriented representation. Both of them assume a string of integers and a string length equal to the number of tasks to be proceeded in the assembly line. If the i-th position of the string has the value j, then, using the station-oriented representation, task i is assigned to workstation j. While, using the task-oriented representation, task j in location i of the string will be assigned to a workstation before the task in location $(i+1)$ of the string. The tasks are allocated into stations starting from the first $(k=1)$ and considering the other stations successively. When a station is loaded maximally, it is closed, and a new station $(k+1)$ is opened. A solution is feasible when the generated sequence of the tasks in the line does not break the specified precedence constraints. In this paper we used task-oriented approach. Also Nearchou [25] states that task-oriented approach is superior in both speed of convergence and quality of solution. This mechanism is implemented using the following procedure proposed by Nearchou [25]:

Begin
For all single solution **do**
Set U=Φ;
Repeat
For all $i \in V$ **do**
If i has no predecessors **then** $U = U \bigcup \{i\}$, i.e. insert i into the set U;

Determine the gene ψ_i of ψ with the maximum value for all $i \in U$

Insert task i into the next available position in the partial schedule (PS)

$U = U/\{i\}$, i.e. remove task i from U;

Until *PS* has been completed;

Return *PS*

End

3-4. How a DEA's phenotype is decoded into a SAL re-balancing problem solution?

Once a specific floating-point vector (i.e., a DEA's genotype) is encoded into a feasible assembly line re-balancing solution (DEA's phenotype), an appropriate decoding scheme is needed to map this phenotype to an actual solution for SAL re-balancing problem. In other words, a method is needed to assign the tasks in the generated task sequence into the workstations. The main idea is to face SAL re-balancing problem through an iterated procedure that solves the corresponding SALBP-1 with a cycle time value being progressively decreased until reaching a near-optimum value within a specific permitted range. This decoding scheme is as follows:

Pseudo code of assigning the task to station and calculating cycle time (Topological-ordering-encoding):

Begin

Set $\bar{c}$=*tsum/m;*

While *cw*<$\bar{c}$ **do**

Assign as many tasks as possible into the first (*m-1*) workstations;

Assign all the remaining tasks to the last workstation, *m*;

For *z*=1 **to** *m* **do**

Calculate *Wz=tSz;*

For *z*=1 **to** (*m-1*) **do**

Calculate *PWz= tSz+* the processing time of the first task assigned to (*z+1*)st station;

Set *cw*=**max** $\{W_1, W_2, ..., W_m\}$ and $\bar{c}$=**min** $\{PW_1, PW_2, ..., PW_{m-1}\}$;

Return *cw*

End

3-5. Evaluation Mechanism

To investigate the quality of each solution, a mechanism corresponding to the fitness function for each phenotype is needed. To this end, a fitness function is defined as the performance evaluation of the solutions. Fitness is usually represented with a unique function that indicates relative superiority of solutions. As mentioned before, in this study the following objectives are to be minimized: the incurred costs and (ii) non-smoothing of reconfigured line. The weighted sum of objectives is suggested as follows.

$$Fitness = w_1 f_1 + w_2 f_2$$

in which

$$f_1 = \frac{NSS[i]}{NSS^*} \ , \ f_2 = \frac{SX^*}{SX[i]} \qquad (13)$$

and w_1 and w_2 are relative importance of f_1 and f_2 respectively, and can be set upon requirements. The objective f_1 in this problem is to maximize number of tasks which assign to their previous stations given a fixed station number *m* and predetermined cycle time based on sailing factors while the objective f_2 is non-smoothing of reconfigures line. Hence the following fitness function is suggested:

NSS= number of tasks which assign to their previous stations

Pseudo code of calculating objective function is as follows.

Begin

For *i*=1 **to** *N* **do**

If *St*[*i*] =St_1 [*i*] **then** *NSS*=*NSS*+1;

End

In above procedure, *St*[*i*] presents the station of i^{th} task while St_1 [*i*] presents the previous station of i^{th} task.

4. Taguchi Experimental Design

The DEA is easy to implement with fast convergence, and requires a few control parameters. It generally shows reliable, accurate and fast performance. However, the user is required to set the values of control parameters for each problem which is a time consuming procedure. To achieve better robustness of the algorithm by not producing functional variance under external environment influence, the ''parameter design'' developed by Dr. Taguchi in early 1960s can be applied. Taguchi discussed that the optimal operator combination is to

minimize variances of quality characteristics resulted from *S/N* ratio, which explains the reason why parameter design is also called robust design. As well as *S/N* ratio which is utilized for minimizing the variances, the mean of quality characteristics is also used for determining the adjustment factors which are utilized for causing to approach the quality characteristic to the objective point. Additionally, quality characteristic of this research is expected *NSS*, which prefers the higher is better principle.

4-1. Generation of test data and selection of Taguchi scheme

An experiment was conducted to test the performance of algorithm. Parameters required for the algorithm are crossover rate (*CR*), scaling parameters (*Fs*), rate (coefficient) of best function (*r*), penalty (*PR*) and finally pop-size and iterations (*NP&NG*). Factors and their levels are shown in Table 1.

Tab. 1. Factor levels

Factor	*Index of level*	*Level*
CR	1	0.5
	2	0.75
	3	0.9
	4	1
Fs	1	0.75
	2	0.9
	3	1
	4	1.5
r	1	0
	2	0.3
	3	0.5
	4	0.8
(NP, NG)	1	$(n, 3n)$
	2	$(1.75n, 1.7n)$
	3	$(2n, 1.5n)$
	4	$(3n, n)$
PR	1	10
	2	12
	3	14
	4	16

Data required for a problem consist of the precedence graph, new cycle time, and primary assignment. The precedence graphs are available at [31]. The performance of DEA was examined over a large set of benchmark ALB instances taken from the open literature and randomly generated primary assignment and new cycle time lower than current cycle time. The benchmarks include test instances organized into two data sets. Data set 1 contains 128 instances for 9 precedence graphs with tasks varying from 29 to 111, while data set 2 contains 174 instances concerning 8 precedence graphs with tasks varying from 53 to 297. In this study we integrate all instances and classified them into three sets. First set contains precedence graph with task varying from 29 to 44 (small size problems), second set contains precedence graph with task varying from 45 to 69 (medium size problems) and finally third sets contains precedence graph with higher 70 tasks (large size problems). Considered benchmarks are summarized in Table 2.

Tab. 2. Considered Benchmarks

Problem size	*Processing graph*	*Number of stations(m)*
Small	Buxey	7
		9
		10
		12
		14
Medium	Warnecke	5
		9
		13
		17
		21
Large	Tonge	5
		10
		15
		20
		25

The full factorial experiment design for aforesaid five factors requires 4^5 experiments. We used fractional replicated designs. For

selecting appropriate orthogonal array it is required calculating the number of degree of freedom. In this research, a degree of freedom for total mean and three degree of freedom for each factor with three levels ($3 \times 5 = 15$) are required. Thus, sum of required degree of freedom equals to: $1 + 3 \times 5 = 16$. Therefore, the appropriate array at least must have 16 rows. Table 3 shows the orthogonal array $L_{16}(4^5)$, where control factors are assigned to the columns of the orthogonal array.

Tab. 3. The Orthogonal Array $L_{16}(4^5)$

Trial	CR	Fs	r	(NG, NP)	PR
1	1	1	1	1	1
2	1	2	2	2	2
3	1	3	3	3	3
4	1	4	4	4	4
5	2	1	2	3	4
6	2	2	1	4	3
7	2	3	4	1	2
8	2	4	3	2	1
9	3	1	3	4	2
10	3	2	4	3	1
11	3	3	1	2	4
12	3	4	2	1	3
13	4	1	4	2	3
14	4	2	3	1	4
15	4	3	2	4	1
16	4	4	1	3	2

5. Experimental result

As mentioned above, the performance of DEA was examined over a large set of benchmark ALB instances taken from the open literature and 3 sets of data were considered for small, medium, and large size problems. For each set, differential evolution was experimented based on the orthogonal array distribution method, so 16 different level combinations of control factors were considered. For each trial in a set, five instances were randomly considered and four replication were performed. After transforming the obtained data into S/N values, the S/N ratio results of expected NSS are summarized in Table 4.

Tab. 4. The S/N Ratios for Expected NSS

Trial	*Control Parameter*					*S/N Ratio*		
	CR	Fs	r	(NG, NP)	PR	Small	Medium	Large
1	1	1	1	1	1	26.526	31.172	33.044
2	1	2	2	2	2	26.902	31.476	33.416
3	1	3	3	3	3	27.171	31.849	33.851
4	1	4	4	4	4	26.848	31.878	34.557
5	2	1	2	3	4	26.796	31.591	33.416
6	2	2	1	4	3	26.269	30.892	32.272
7	2	3	4	1	2	26.555	31.778	33.752
8	2	4	3	2	1	26.822	31.860	33.962
9	3	1	3	4	2	26.991	32.056	34.195
10	3	2	4	3	1	26.268	31.614	33.676
11	3	3	1	2	4	26.238	30.886	32.730
12	3	4	2	1	3	26.480	31.828	34.109
13	4	1	4	2	3	25.334	31.045	32.363
14	4	2	3	1	4	25.819	32.114	33.405
15	4	3	2	4	1	26.537	31.691	33.679
16	4	4	1	3	2	26.049	31.381	32.669

The best robustness of the algorithm are shown in Table 5.

Tab. 5. Optimized Factor Levels

Factor	Optimized level		
	Small	Medium	Large
CR	0.5	0.5	0.9 &
Fs	1	1.5	1.5
r	0.5	0.5	0.5
(NP, NG)	$(3n, n)$	$(n, 3n)$	$(3n, n)$
PR	12	12	10

For understanding about adjustment factors, we use the mean of relative percentage deviations (*RPD*) for *NSS*. The *RPD* value is defined as follows:

$$RPD_{ij} = \frac{max_j(NSS_{ij}) - NSS_{ij}}{max_j(NSS_{ij})} \qquad (14)$$

i and *j* denote respectively index of trial and replication. After transforming the obtained data into *RPD* values, the mean *RPD* results of expected *NSS* are summarized in Table 6.

Tab. 6. The Value Results for Expected NSS

Trial	*Control Parameter*					*Expected RPD*		
	CR	Fs	r	(NG,NP)	PR	Small	Medium	Large
1	1	1	1	1	1	0.163	0.212	0.288
2	1	2	2	2	2	0.098	0.225	0.247
3	1	3	3	3	3	0.112	0.203	0.226
4	1	4	4	4	4	0.071	0.195	0.163
5	2	1	2	3	4	0.137	0.231	0.255
6	2	2	1	4	3	0.125	0.212	0.302
7	2	3	4	1	2	0.102	0.186	0.217
8	2	4	3	2	1	0.140	0.196	0.207
9	3	1	3	4	2	0.125	0.203	0.193
10	3	2	4	3	1	0.129	0.197	0.228
11	3	3	1	2	4	0.121	0.219	0.295
12	3	4	2	1	3	0.100	0.205	0.204
13	4	1	4	2	3	0.172	0.188	0.284
14	4	2	3	1	4	0.223	0.151	0.227
15	4	3	2	4	1	0.096	0.211	0.217
16	4	4	1	3	2	0.208	0.179	0.277

Here in this section we implement some additional computational experiments. The proposed solution approaches which are omitting approach and taking penalty of infeasible solution during DEA were implemented in Delphi7. The machine used was a PC with P4 CPU processors of 2.8 GHz running with 256 MB of RAM. To measure the effectiveness of the approaches, we compared the performance of the two approaches against each other. An experiment was conducted to test the performance of the solution approaches. Parameters required for the algorithm consists of the crossover rate (*CR*), scaling parameters (*Fs*), rate of best function (*r*), penalty (*PR*) and finally pop-size and iterations (*NP&NG*). Data required for a problem consist of the precedence graph, new cycle time, and primary assignment. The precedence graphs are available at http://www.assembly-line-balancing.de/. To produce a primary assignment, a feasible solution is randomly generated and a new cycle time is considered less than its current cycle time. The benchmarks include test instances organized into two data sets. In this study we integrate all instances and classified them into three sets. First set contains precedence graph with task varying from 29 to 44 (small size problems), second set contains precedence graph with task varying from 45 to 69 (medium size problems) and finally third sets contains precedence graph with higher 70 tasks (large size problems). In small, medium and large size problems 10, 10 and 20 problems have been solved respectively and for each of them 5 iterations of running algorithm have been executed. We compared the performance of the approaches using three measures: average of objective function (*OF*), average percentage error (Error), and standard deviation (Std). The percentage error is defined as 100*(*OF* of the heuristic − OF of the best heuristic)/ (*OF* of the best heuristic). The results of the computational experiments are summarized in Tables 7. Additionally to further analyze the performance of suggested algorithm, its performance is depicted in different size of problems in Figs. 4, 5 and 6.

Tab. 7. Comparison of Penalty and Omitting Approach

Problem size	Problem name	*Penalty approach*			*Omitting approach*		
		Error	OF	Std	Error	OF	Std
Small	buxey,m=7	0.04	24	1	0.08	23	1
	buxey,m=9	0.03	24.2	0.2	0.06	23.4	0.8
	buxey,m=10	0.04	22	0.5	0.09	21	1.5
	buxey,m=12	0.01	23.8	0.2	0.06	22.6	1.3
	buxey,m=14	0.04	22	0.5	0.11	20.4	0.3
	sawyer,m=7	0.04	23	0.5	0.09	21.8	1.7
	sawyer,m=9	0.02	25.4	0.3	0.08	23.8	0.7
	sawyer,m=11	0.02	24.6	0.3	0.06	23.6	1.3
	sawyer,m=14	0.02	23.6	0.3	0.05	22.8	1.2
	gunther,m=10	0.01	30.8	0.2	0.10	27.8	1.7
medium	warneke,m=9	0.01	49.4	0.3	0.03	48.6	0.3
	warneke,m=13	0.02	44.2	0.7	0.03	43.6	2.3
	warneke,m=17	0.00	40.8	0.2	0.03	39.6	1.3
	warneke,m=21	0.01	40.4	0.3	0.08	37.8	4.7
	warneke,m=25	0.01	40.4	0.3	0.06	38.4	7.8

Problem size	Problem name	*Penalty approach*			*Omitting approach*		
		Error	OF	Std	Error	OF	Std
	warneke,m=29	0.02	39.4	0.3	0.06	37.8	2.7
	kilbridge,m=4	0.02	39.2	0.7	0.07	37.4	4.3
	kilbridge,m=6	0.02	41	1	0.03	40.6	2.8
	kilbridge,m=8	0.01	40.6	0.3	0.08	37.6	3.3
	kilbridge,m=10	0.02	40	1	0.04	39.4	1.3
Large	tonge,m=5	0.02	66.4	1.3	0.04	65	0.5
	tonge,m=8	0.02	63	1	0.03	61.8	0.7
	tonge,m=10	0.00	62	0	0.01	61.4	0.8
	tonge,m=12	0.02	62	1	0.03	60.8	0.7
	tonge,m=15	0.00	59.8	0.2	0.02	58.6	0.8
	tonge,m=18	0.01	57.6	0.3	0.02	57	0.5
	tonge,m=20	0.01	53.4	0.3	0.03	52.4	1.3
	tonge,m=22	0.01	49.4	0.3	0.04	48	1.5
	tonge,m=24	0.02	57	0.5	0.04	55.6	1.3
	tonge,m=25	0.01	51.4	0.3	0.03	50.4	0.8
	lutz3,m=4	0.01	83.4	0.8	0.07	78.2	3.2
	lutz3,m=9	0.01	80.4	0.3	0.04	77.4	6.8
	lutz3,m=13	0.01	78	1	0.04	75.8	4.2
	lutz3,m=17	0.01	75.2	0.7	0.05	72.2	9.2
	lutz3,m=21	0.01	74.6	0.3	0.05	71.6	4.3
	lutz2,m=10	0.04	69.4	1.3	0.06	67.6	22.3
	lutz2,m=12	0.01	71.2	0.7	0.06	67.8	12.7
	lutz2,m=16	0.01	73	0.5	0.06	69.6	10.8
	lutz2,m=20	0.03	68.8	1.2	0.06	66.6	8.3
	lutz2,m=24	0.06	67.8	2.7	0.05	68.2	7.2

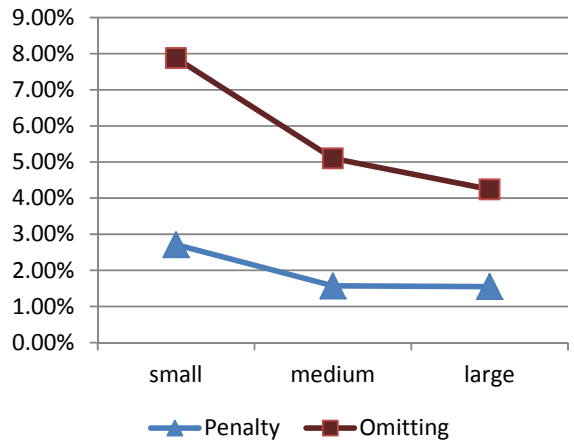

Fig. 1. Overall error for different size of problem

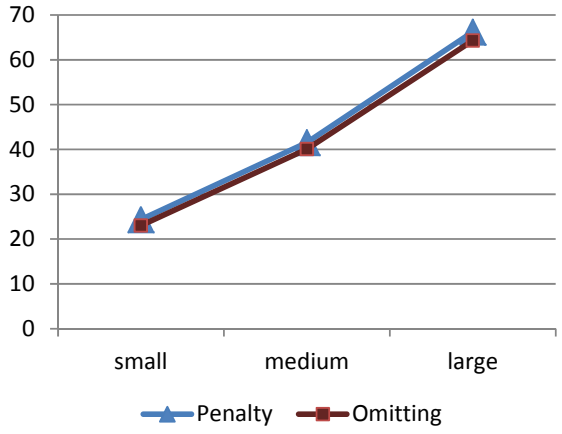

Fig. 2. Overall *OF* for different size of problem

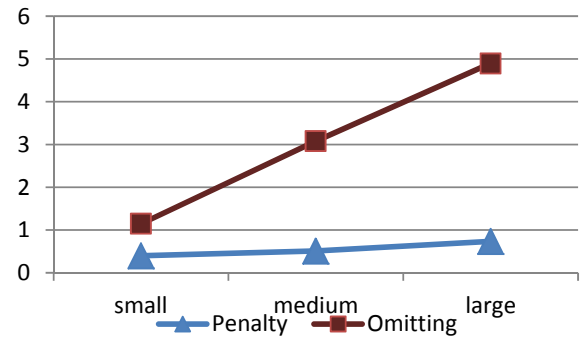

Fig. 3. Overall Std for different size of problem

As indicated in Figures 1, 2 and 3, penalty approach performs much better than omitting one. Figure 1 and 3 indicate that the error and Std of penalty are smaller than those of omitting, while Figure 2 indicates that value of *OF* for penalty approach is larger than omitting one.

5. Conclusions and Future Study

The differential evolution algorithm (DEA) as a well known population based stochastic algorithm used to solve the problem of assembly line re-balancing. Due to the cost of evaluating real-world functions, optimization algorithms must be both fast and accurate. DEA is easy to implement and requires a few control parameters. It generally shows reliable and accurate performance. Hence, it was addressed in order to re-balancing an existing line as similar as previous one. Despite of many advantages of DEA, like other nature inspired algorithms, it suffers from the cost of parameter setting. To overcome these issues and improving the convergence speed of DEA, a detailed statistical experiment based on Taguchi experimental design is performed. Moreover, two approach were considered to address infeasibility occurred during the solving procedure: (i) Omitting infeasible solution, and (ii) Considering penalty for the objective function of infeasible solution. After experimenting both schemes over selected test beds (from a set of benchmarks described), we found that penalty method is superior than first one in terms of speed of convergence and quality of solutions; therefore, it has been decided to adopt this scheme to DE. After developing differential evolution algorithm to minimize the differences between previous and re-balanced line, we tried to find optimized DEA parameter levels and discussed calibration of DEA controlling parameters.

References

[1] Salveson, M.E.,. The assembly line balancing problem. Journal of Industrial Engineering. 6, (1955), pp 18–25.

[2] Scholl, A., Becker, C., A note on an exact method for cost oriented assembly line balancing. International Journal of Production Economics 97, (2005),. pp 343 –352.

[3] Boysen, N., Fliedner, M., Scholl., A. Assembly line balancing: Which model to use when? International journal of Production Economics 111 (2008), pp 509–528.

[4] Akagi, F., Osaki, H., Kikuchi, S., A method for assembly line balancing with more than one worker in each station. International Journal of Production Research 21, (1983), pp 755–770.

[5] Wilhelm, W.E., Gadidov, R., A branch-and-cut approach for a generic multiple-product, assembly-system design problem. INFORMS Journal on Computing 16, (2004), pp 39–55.

[6] Ghosh, S., Gagnon, R.J., A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems. International Journal of Production Research 27, (1989), pp 637–670.

[7] Erel, E., Sarin, S.C., A survey of the assembly line balancing procedures. Production Planning & Control 9, (1998), pp 414–434.

[8] Rekiek, B., Design of assembly lines. Me´ moire pre´ sente´ en vue de l'obtention du grade de docteur en sciences applique´ es. Universite´ libre de Bruxelles, Brussels, Belgium. (2000).

[9] Schofield, N.A., Assembly line balancing and the application of computer techniques. Computers and Industrial Engineering 3, (1979), pp 53–59.

[10] Falkenauer, E., 2005. Line balancing in the real world. In: Proceedings of the International Conference on Product Lifecycle Management PLM 05, Lumiere University of Lyon, France, (2005).

[11] Gamberini, R., Grassi, A., Rimini, B., A new multi-objective heuristic algorithm for solving the stochastic assembly line re-balancing problem. International journal of Production Economics 102, (2006), pp 226–243.

[12] Gamberini, R., Grassi, A., Gamberi, M., Manzini, R., Regattieri, A., U-shaped assembly lines with stochastic tasks execution times: heuristic procedures for balancing and rebalancing problems. In: Proceedings of the Business and Industry Symposium, (2004) Advanced Simulation Technologies Conference, Arlington, Virginia. (2004).

[13] Bautista, J., Pereira, J., Ant algorithms for a time and space constrained assembly line balancing problem. European Journal of Operational Research 177, (2007), pp 2016–2032.

[14] Johnson RV., Assembly line balancing algorithms: computation comparisons. Int J Prod Res 19, (1981), pp 277–287.

[15] Hoffmann TR., EUREKA: a hybrid system for assembly line balancing. Manage Sci 38, (1992), pp 39 –47.

[16] Jackson JR., A computing procedure for a line balancing problem. Manage Sci, 2, (1956), pp 261 – 271.

[17] Schrage LE, Baker KR Dynamic programming solution of sequencing problems with precedence constraints. Oper Res 26, (1978), pp 444– 449.

[18] Scholl A, Becker C., State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. Eur J Oper Res 168, (2006), pp 666 –693.

[19] Karp RM Reducibility among combinatorial problems. In: Miller RE, Thatcher JW (eds) Complexity of computer applications. Plenum, New York, (1972).

[20] Baybars I A survey of exact algorithms for the simple assembly line balancing problem. Manage Sci, 32, (1986), pp 909–932.

[21] Andreas C. Nearchou Balancing large assembly lines by a new heuristic based on differential evolution method. Int J Adv Manuf Technol, 34, (2007), pp 1016–1029.

[22] Kirkpatrick S, Gelatt CD Jr, Vecchi MP Optimization by simulated annealing. Science, 220, (1983), pp 671– 680.

[23] Glover F., Tabu-search-Part I. O RS A J Comput, 1, (1989), pp 190–206.

[24] Glover F., Tabu-search-Part II. ORSA J Comput, 2, (1990), pp 4 –32.

[25] A. C. NEARCHOU. Multi-objective balancing of assembly lines by population heuristics, International Journal of Production Research, 46, (2008), pp 8, 15.

[26] Storn R, Price K., Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim, 11, (1997), pp 341–354.

[27] Ali MM, Törn A (2004) Population set-based global optimization algorithms: some modifications and numerical studies. Comput Oper Res, 31, (2004), pp 1703–1725.

[28] Kaelo P, Ali MM., A numerical study of some modified differential evolution algorithms. Eur J Oper Res, 171, (2006), pp 674–692.

[29] Nearchou AC., Meta-heuristics from nature for the loop layout design problem. Int J Prod Econ, 101, (2006), pp 312–328.

[30] Onwubolu GO, Davendra D Scheduling flow shops using differential evolution. Eur J Oper Res, 169, (2006), pp 1176–1184.

[31] Homepage for assembly line optimization research, available from: http://www.assembly-line-balancing.de/.