Sequencing Mixed Model Assembly Line Problem to Minimize Line Stoppages Cost by a Modified Simulated Annealing Algorithm Based on Cloud Theory

Zaman Zamami Amlashi[a,*], Mostafa Zandieh[b]

[a] *MSc, Faculty of Industrial and Mechanical Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran*

[b] *Assistant Professor, Department of Industrial Management, Management and Accounting, Shahid Beheshti University, Tehran, Iran*

Received 19 October, 2010; Revised 13 November, 2010; Accepted 25 December, 2010

Abstract

This research presents a new application of the cloud theory-based simulated annealing algorithm to solve mixed model assembly line sequencing problems where line stoppage cost is expected to be optimized. This objective is highly significant in mixed model assembly line sequencing problems based on just-in-time production system. Moreover, this type of problem is NP-hard and solving this problem through some classical approaches such as total enumeration or exact mathematical procedures such as dynamic programming is computationally prohibitive. Therefore, we proposed the cloud theory-based simulated annealing algorithm (CSA) to address it. Previous researches indicates that evolutionary algorithms especially simulated annealing (SA) is an appropriate method to solve this problem; so we compared CSA with SA in this study to validate the proposed CSA algorithm. Experimentation shows that the CSA approach outperforms the SA approach in both CPU time and objective function especially in large size problems.

Keywords: Sequencing problem; Mixed-model assembly line; Just-in-time production system; Cloud theory; Simulated annealing; Minimizing line stoppages.

1. Introduction

This research presents a new application of simulated annealing algorithm based on cloud theory (CSA) in order to address mixed model assembly line sequencing problem in a just-in-time (JIT) production system. A mixed model assembly line (MMAL) is a type of production lines where a variety of products having similar characteristics are assembled. The effective utilization of a mixed-model assembly line requires solving two problems in a sequential manner: 1) line design and balancing, and 2) determination of the production sequence for different models. Determining the balance and the sequence of products to be produced on MMALSPs has received considerable attention from researchers. In this work, we focus on the mixed-model sequencing problem assuming that line balancing has already been achieved using an average cycle time for the products to be assembled. This problem is known as the Mixed-Model Assembly Line Sequencing Problem (MMALSP). To validate the proposed algorithm, we compare CSA with simulated annealing algorithm (SA) that is known as an effective algorithm to consider in MMALSP. The comparison of two algorithms is based on two groups of problems; small and large size problems that are designed for structure of line stoppages cost objective function. SA is a Monte Carlo based optimization approach that simulates the physical annealing process to find the optimal solution Laarhoven et al. [14]. Temperature is descended gradually at each step and it is constant in each stage until the stop criteria is satisfied, the resulted solutions are acceptable if they are improved; otherwise it will be accepted with a certain probability.

Cloud theory is a model in the fuzzy theory which is related to quality concepts and quantity data, Deyi et al. [4]). We used cloud theory-based on Metropolis law to produce a group of continuous temperatures close to a fixed temperature. Manufacturing random temperatures based on

*Corresponding author E-mail: zamami62@yahoo.com

cloud theory can preserve diversity. In Section 2, literature review on MMALSP, SA and cloud theory will be presented. Section 3 presents a description of the MMALSP problem and will formulate line stoppage cost objective. Section 4 will introduce the proposed cloud theory-based simulated annealing algorithm. Section 5 illustrates the computational results which compared the results of CSA with SA in two groups of small-medium and large size problems. Finally Section 6 provides conclusions and suggestions for future research.

2. Literature Review

Following the first study on mixed model assembly line conducted by Kilbridge and Webster [10], several researchers studied the MMALSP. Thomopoulos [30], Dar-El and Cother [2], Dar-El and Cucuy [3] addressed minimizing line length goal and offered an integer programming and heuristic procedure to resolve this problem. Okamura and Yamashina [26] offer a heuristic method to consider minimizing line stoppage objective function. These studies have different objectives such as minimizing line length or line stoppages, but their common feature is that they all address the final assembly line and ignore other levels in the multi-level production systems. The first analyses of mixed-model, multi-level production systems were made by Monden [25], and Miltenburg and Sinnamon [23]. They consider multi-level model production systems to solve the same problem with the objective of keeping a constant rate of supply of every part used by the system. Miltenburg et al. [24] studies the mixed-model sequencing problem by considering the variation in production rates of the finished products. He developed a dynamic programming and also two heuristics to solve the problem. In a follow-up study, Miltenburg and Sinnamon [23] consider the same problem and propose heuristic procedures to solve large-sized problems. Kubiak and Sethi [13] studied the MMALSP with the objective of minimizing product-usage variation. They developed an assignment model to generate optimal schedules for MMALs. Bard et al. [1] developed a model involving two objectives; minimizing the overall line length and keeping a constant rate of part usage. They solved the problem by using the weighted sum and proposed a tabu search (TS) method. Hyun et al. [9] addressed three objectives: minimizing total utility work, keeping a constant rate of part usage, and minimizing total setup cost. This problem was solved by proposing a new genetic evaluation and selection mechanism. They named this new algorithm by PS-NC GA. McMullen [16] considered two objectives: minimizing the number of setups and keeping a constant rate of part usage, and solved this problem by using the weighted sum and TS method. Korkmazel and Meral [12] developed the weighted-sum approach for two goals introduced by Monden [25]. McMullen and Frazier [17] developed a simulated annealing (SA) method for the model used by McMullen [16] and compared it to the Tabu Search method. McMullen et al. ([18, 19, 20]) also solved the same problem using genetic algorithms (GAs), Kohonen self organizing map (SOM), and ant colony optimization (ACO), respectively, and compared their performance with SA and TS methods. Mansouri [11] also solved the same problem using genetic algorithms. He introduced a new selection mechanism and in small size problems where produced by McMullen [16] used a total enumeration (TE) to find pareto-optimal frontier. Tavakkoli-Moghaddam and Rahim-Vahed [28] solved the problem proposed by Hyun et al. [9] by using a new memetic algorithm. McMullen and Tarasewich [21] considered the problem of mixed-model sequencing through setups. They developed a beam search heuristic to generate efficient frontiers. Their experimental results indicate that the proposed approach performs well in terms of both solution quality and computation times.

3. Mixed Model Assembly Line Sequencing Problem

Mixed-model assembly lines are flow oriented production systems. These production lines are capable of responding customer requirements when their demands change continually. For a variety of models, withholding low inventories are needed for successful performance of JIT production systems. The effective utilization of a mixed-model line requires that managers address two key problems: Miltenburg & Sinnamon [23]:

(i) The assignment of tasks to workstations (i.e. the line balanc-ing problem).

(ii) The sequencing of models on the line (i.e. mixed model sequencing problem).

In this work, we focus on the mixed-model sequencing problem assuming that line balancing has already been achieved using an average cycle time for the products to be assembled.

Descriptions of some terminologies for the lines are presented below:

(1) Launch interval: is defined as a fixed-time at interval which successive work-pieces are fed into a station. There are two types of launching intervals: (a) fixed rate launching in which the launching period is a weighted average of the total assembly time for all products to be assembled over all stations, in this case production cycle time must be less than or equal model cycle time; and (b) Variable rate launching in which the launching period is the task time of the last product launched at the first station so the worker at the first station can start working on the next product immediately after completing work on the current product. When units are launch at variable rates and in an arbitrary order, it is necessary for optimum utilization of the line, that worker cycle time must be equal to maximum model cycle time of unit, Kilbridge and Wester [10].

(2) Starting point of work: the starting point of workplace in a station is upstream position where a worker starts working on it. This point may be either on the zero reference point or within the station in a closed-station system, and may be within or before the zero reference point in an open-station system.

(3) Minimal part set (MPS): in this research, a minimal part set is defined as the smallest possible set as parts in the same proportion as the demands mix during the whole working period. Suppose the model A, B, and C have the demand of 500, 300, and 400 units, respectively, then, it is difficult to sequence a total of 1200 or more work-piece at one time. This demand set {500,300,400} is divided by its largest common divisor (which is 100 in this case) to obtain the minimal part set a {5, 3,4}. The problem of scheduling all products during the working period is then reduced to finding the assigned order of models to stations in minimum part sets {5, 3, 4} order. The number of times that a minimal part set repeats in order to complete the demand during and the entire working period is the largest divisor, called frequency, F. In this case, F = 100. Motivations for working with the minimum part set are as follows: First, it is becoming common practice in industry to plan for production in terms of the minimum part set, especially in flexible manufacturing. Second, the approach greatly simplifies the computations, thereby permitting the derivation of optimal solution for problems of realistic size. Third, the results obtained from working with the minimum part set MPS rather than the full part is surprisingly better. Dar-EL and Cother [2], Thomopoulous [30].

(4) Closed station: In closed station, the worker must work within the limits of the station; the worker is not allowed to move out of his work limit area when he assembles the product. This happens, for instance, in pits or in paint booths where the assigned work cannot be accomplished outside.

3.1. Assumptions

Different products with similar production characteristics are produced on the assembly line which has a finite number of workstations. a conveyor speed has a constant rate v_c (in this survey $v_c = 1$). Figure 1 shows a conveyor belt with different products. Similar products are launched onto the conveyor at a constant rate γ. The line is partitioned into J stations. It is assumed that the stations are all of closed types. A closed station has boundaries that workers cannot cross. Such a closed station is often found in reality where the use of facilities is restricted within a certain boundary. The tasks allocated to each station are properly balanced and their operating times are deterministic. The worker moves downstream on the conveyor while performing his/her tasks to assemble a product. To complete the job, the worker moves upstream to the next product. The travel times of workers are ignored. There are two types of worker schedules, early start schedule and late start schedule in the literature. An early start schedule is more common in practice and is used in this survey, Hyun et al. [9]. The bulk of literature confirms that setup times between different products are negligible. MPS is a vector representing a product mix, such that $(d_1,...,d_a) = (D_1/H,...,D_a/H)$ where a the total number of models is, $D_i\ (i = 1,...,a)$ is the number of products of model type i that needs to be assembled during the entire planning horizon and H is the greatest common divisor or the largest common factor of $D_1, D_2,..., D_a$. This strategy operates in a cyclical manner. Assuming that the total demand for products is *H* and demand for models is a in the planned period, obviously, *H/D* times the repetition of producing the MPS products can meet the total demand in the planning horizon.

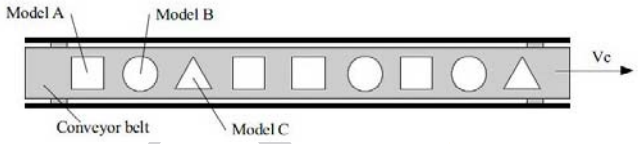

Fig. 1. A Diagram of assembly line conveyor belt in closed station

Other assumptions are considered as follows:

1) The assembly line is a closed station ones;
2) No buffer exists between any two adjacent stations;
3) The layout of the mixed-model line assembly system is one dimensional;
4) The worker must work within the work zone assigned to him;
5) A work-piece is loaded on the conveyor belt every l minute $(\gamma = 1)$;
6) Work-pieces are fixed on the moving belt;
7) There is no more than one work-piece at a station at a time;
8) Each station has only one worker;
9) Stations in the system are connected by a moving conveyor;
10) Workers perform at a constant rate in either direction of movement;
11) Maximum length of assembly line is given.

Figure 2 illustrates the construction of mixed model assembly time for an example with MPS = (B,C,B,A,B,A).

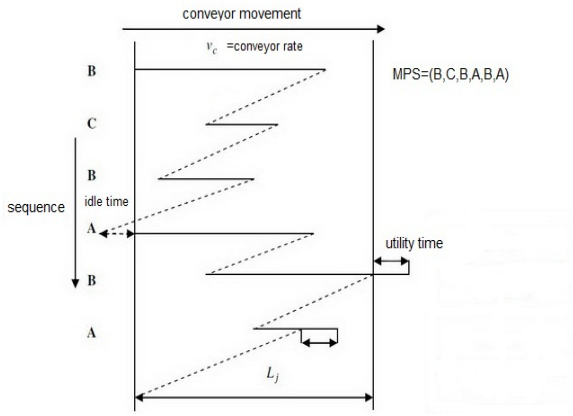

Fig. 2. Diagram of operations in a closed workstation (Rahimi vahed et al [29])

3.2. Minimizing Line Stoppage Cost

Initially, Okamora and yamashina [26] presented a model to minimize line stoppages and proposed a heuristic reduction, maximum point of completion time of models operation. One of the most important objectives is line stoppage in the actual and required capacity of the line which affects the stoppage of conveyer. This objective specifies a sequence for minimizing the cost of the line stoppages which changes in each sequence for part shortage of a model, enabling a good setup, etc. In fact, this objective considers the capacity of the line without using utility workers, bad condition for producing a model like the time of arriving the part of each model in each sequence.

To minimize line stoppage cost Monden [25] maintain that one must avoid consecrating several models with large operation time. It is appropriate, after each model with large operation time, that one model with small operation time be allocated to the next position in sequence.

Before presenting the mathematical representation of line stoppage cost goal, the following notations are defined:

J Number of workstations

a Number of unique products to be produced

DT Total number of units for all products – also represents number of positions in sequence-

The number of products produced in one cycle is given by:

$$DT = \sum_{i=1}^{a} d_i \tag{1}$$

d_i Demand for product $i = 1,2,...,a$

t_{ij} Operation time for one unit of model i in station j

L_j Length of workstation j

α'_j Coefficient of variation of stations when line stoppages accurate that determinate as follow:

$$\alpha'_j = \frac{\max\{t_{ij}\}}{L_j} \tag{2}$$

$\overline{T_j}$ Determination of the operation time average for one unit of product in work station j :

$$\overline{T_j} = \frac{\sum_{i=1}^{a} d_i t_{ij}}{DT} \tag{3}$$

k Counter of positions in sequence

$X_{i,k}$ Total number of units of products i produced over stages 1 to k

Mathematical formulation of this goal is presented below:

$$\min \sum_{j=1}^{J} \alpha'_j \sum_{k=2}^{DT} \left| \sum_{i=1}^{a} t_{ij} \left(X_{ik} + X_{i(k-1)} - 2\overline{T_j} \right) \right| \tag{4}$$

$$s.t:$$

$$\sum_{k=1}^{DT} X_{ik} = d_i \quad i = 1,2,...,a \tag{5}$$

$$\sum_{i=1}^{a} X_{ik} = 1 \quad i = 1,2,...,a \tag{6}$$

$$X_{ik} \in \{0,1\} \quad i = 1,2,...,a \quad k = 1,2,...,DT$$

Equation (4.1) ensures that total demands of model are satisfied based on MPS in the planning horizon period. Equation (4.2) ensures that each position in sequence is used with one unit of products.

3.3. Combinatorial Complexity

The number of feasible solutions increases exponentially, when the problem size increases. To find production sequences with desirable levels of all objectives NP-hard as pointed by Hyun et al. [9] is used. The total number of sequences for a MMALSP can be computed as follows:

$$Total\ sequence = \frac{(\sum_{i=1}^{a} d_i)!}{\prod_{i=1}^{a} (d_i!)} \tag{7}$$

Thus, problems with a large number possible solution usually cannot be solved optimality within a reasonable amount of time, Mansouri [15].

4. The proposed Cloud Theory-Based Simulated Annealing Algorithm

4.1. Simulated Annealing Algorithm in General

Simulated annealing algorithm (SA) is a stochastic optimization method that is based on iterative strategy. SA can generate near-optimal solutions to combinatorial optimization problems. Kirkpatrick [11], Eglese [7] and Goldberg [8] provide fundamental descriptions of simulated annealing in addition to informative examples. Simulated annealing has been applied to a vast number of single and multiple objective optimization problems. It has a strong analogy to combinatorial optimization and the physical process of crystallization named annealing, Laarhoven et al. [14]. It is assumed here that the reader is familiar with the basics of Simulated Annealing. SA starts from a high initial temperature, and searches the solution space to find near-optimal solutions randomly using metropolis criterion, and escapes from local optimum trapped through the acceptance of ‘‘bad solution’’ with a certain probability. As the temperature decreases gradually, it repeats the above process and finds the global optimization solution finally. As an effective combinatorial optimization method, SA has been proved to be strict in theory and useful in many application fields.

The core of SA is Metropolis procedure, which simulates the annealing process at a given temperature *T*, Metropolis et al. [22]. This procedure is named after the scientist who devised a similar scheme to simulate a collection of atoms in equilibrium at a given temperature. This improving mechanism starts with a primary temperature T_0 and primary solution s_0, other solutions in the solution space are found in the following way: T_0 is decreased according to the cooling schedule function, reaching thermal equilibrium is time consuming and in this interval another solution s_{new} is found in the neighbourhood of the previous solution s_{curr}. If the value of objective function of neighbourhood solution $f(s_{new})$ is less than the current value in minimization problems, the new solution will be accepted; otherwise, first generate a random number *r* between [0,1] that obeys uniform distributive rule, then the bad solution will be accepted with probability $p = e^{\frac{-\Delta}{T}}$ if $r < p$ Where $\Delta = \frac{f(s_{new}) - f(s_{curr})}{f(s_{new})}$ and *T* is the current temperature. This process is repeated until one of the stopping criteria is satisfied. Laarhoven [14].

4.2. Solution Representation

One of the most widely used representations for mixed-model assembly line sequencing problems is job-to-position representation. In this kind of representation, a single row array of the size equal to the number of all products to be scheduled is considered. The value of the first element of the array shows which job is scheduled first. The second value shows which job is scheduled second and so on. Suppose that an MPS be (ABABCB) that two numbers of product ''A'', three numbers of product ''B'' and one number of product ''C'' must be satisfied. Table 1 shows how this representation is depicted. The operators often applied to real values need to be redefined to make them applicable on a permutation of jobs. Table 1 shows the job-to-position representation.

In so far as the proposed CSA algorithm obeys cloud theory and the concept of fuzzy membership function pin et al. [27] and in order to obtain the real-coded representation from it, in this survey first 6 (the number of the jobs) random numbers between [0,1] are needed to be generated. Afterwards, these numbers will be sorted and the two smallest will be assigned to the first production model, category A; the next three smallest group of them will be assigned to the second production model, category B; and, finally, the last number will be assigned to the third production model, category C. Table 2 demonstrates the resulting real-coded encoding according to the random numbers in Table 3.

Table 1
Job to position representation for a MMALSP

Location in a sequence	1	2	3	4	5	6
job to be scheduled	A	B	A	B	C	B

Table 2
The continuous and real-coded representation of table 1

Location in a sequence	1	2	3	4	5	6
real-coded representation	0.0128	0.563	0.149	0.387	0.742	0.281

Table 3
A sample set of random numbers

n0.1	no.2	no.3	no.4	no.5	no.6
0.149	0.0128	0.387	0.742	0.281	0.563

4.3. Basic Concepts of Cloud Theory

The cloud theory model is created and developed based on membership function in fuzzy theory Di et al. [6]. Thus, transferring between quality concept such as natural language Deyi et al. [5] and quantity data representation, defined as membership function. Let D be the language value of domain v and mapping $M_D(x): v \to [0,1], \forall x \in v, x \to M_D(x)$, if the distribution of $M_D(x)$ is normal, it is named a normal cloud model. This map produces a group of random numbers, with stable tendency that is distinguished by expectation Ex, entropy En and super entropy, He. These three parameters reflect the quantitative features of the concept $M_D(x)$,Deyi et al [5]. Ex determines the centre of the cloud and En determines the range of the cloud. Figure 3, displays that about 99.74% of the total cloud drops distribute between [Ex-3En, Ex+3En]; He determines the degree of cloud drops dispersive, the larger the He is the more dispersive the cloud drops locate, Pin et al. [27]. A normal distribution of Y condition cloud generator (NYCCG) can create a drop of cloud ($drop(x_i, v_0)$) with three parameters Ex, En, He and certain u_0 as follows, Deyi and Yi [5]:

NYCCG operator:

Initiation: *{Ex, En, He}, n,* v_0

For i = 1 to n

$$En' = randn(En, He)$$

$$x_i = Ex \pm En' \cdot \sqrt{-2\ln(v_0)}$$

$$drop(x_i, u_0)$$

End

Output: $\{(x_1, v_0), ..., (x_n, v_0)\}$

Where *randn (En, He)* produces a random number with normal distribution whose expectation is *En* and standard deviation is *He*.

4.4. Cloud Theory-Based Simulated Annealing Algorithm in MMALSP

At first, generate a random group of solutions, and get their standard deviation of line stoppage cost objective function, which is marked as *STD*, then the initial normalization temperature is:

$$T_0 = 1.0 - Arc\cot(STD/\pi) \quad (8)$$

Assume T_F be the final temperature and s_0 be the initial solution. Procedure of the CSA algorithm is given as follow:
Cloud theory based simulated annealing algorithm:

1: Generate 50 solutions randomly and compute *STD* of their objective functions.
2: Let $T_0 = 1.0 - Arc\cot(STD/\pi)$
3: Initialize s_0 randomly
4: $T = T_0$, $s_{curr} = s_0$, $k = 0$, $\lambda = 0.9$
// *k*: is iterations counter //
5: **while** $T > T_F$
/ $T_F = 0.001$ /
6: **while** *metropolis* criteria is satisfied
7: $He = T_k$, $En = T_k$, $u_0 = 1 - T_k$
8: $En' = (En + He - \frac{randn('uniform',0,1)}{3})$
9: $T' = En'\sqrt{-2\ln(u_0)}$
10: $f(s_{curr}) \leftarrow$ Value of objective function with current Solution s_{curr}
11: $s_{new} \leftarrow$ use inversion operator to Generate neighbourhood Solution
// 5 times run inversion method and best solution (Solution with min objective function) is s_{new} //
12: $f(s_{new}) \leftarrow$ Value of objective function with new solution s_{new}
13: **If** $f(s_{new}) \le f(s_{curr})$ **then**
14: $s_{curr} \leftarrow s_{new}$
15: **Else if** $p = e^{\frac{-\Delta}{T}} < rand('uniform'0,1)$ **then**
16: Accept s_{new} and $s_{curr} \leftarrow s_{new}$ where

$$\Delta = \left| \frac{f(s_{new}) - f(s_{curr})}{f(s_{curr})} \right|$$

17: **Else** keep s_{curr} to next stage
18: **End if**
19: *k=k+1*;
20: $T = T_0\lambda^k$;
21: **End 'metropolis' while;**
22: **End 'stopping criteria' while;**

At the high temperature, annealing process has random search tendency and the changeable range of annealing temperature is much wider. When the temperature decreases gradually, particularly at the low temperature, the search process is a goal directed search. The changeable range of annealing temperature is much narrower and less dispersive. So, the annealing process tends towards a stable status. With this explanation, the entropy *En* and the super entropy *He* should be positively correlated with the temperature stage, since *En* determines the range of the cloud and the super entropy *He* determines the cloud drops' dispersive degree in cloud theory, Deyi et al. [4]. So in the proposed algorithm, they are both equal to the temperature *T*.

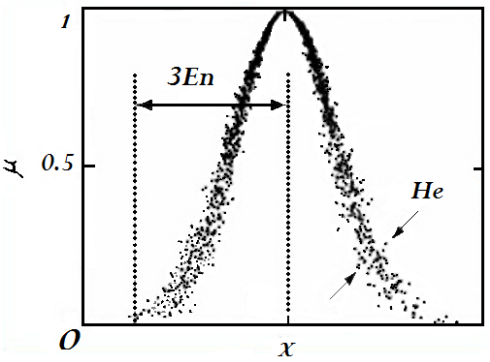

Fig. 3. Three digital characteristics of a normal cloud Pin et al [27]

4.3.1. New Temperature Annealing Process

The annealing temperature in general SA is a certain value in every stage and the searching process is completed between neighbours, but the temperature update function in cloud theory-based simulated annealing algorithm is exponential. Let the stage counter is *k*, then the temperature update function is computed according to the following:

$$T = T_0\lambda^k; k = 1,2,\ldots, 0 < \lambda < 1 \quad (9)$$

In this proposed update function, the temperature is gradually falling. In the CSA, at the end of each stage, temperature changes according to equation (8) as base temperature for next stage. Using a *Y* condition cloud generator and taking base temperature as a certain value produces a group of new values randomly that distribute around the base temperature like a ''cloud". In this generation annealing temperature, the fixed temperature point of each step becomes a changeable temperature zone, the course of temperature changing is nearly continuous and simulates the physical annealing process better as a real world annealing process.

In Figure 4, the CSA backfire and re-annealing temperature process are simulated and performed as a sample problem (the backfire and re-annealing process explained by pin et al. [27]). It is shown that CSA has

continuous temperature based on backfire and re-annealing process and its simulated is so near to physical annealing *process*. But SA temperature is stable for the whole of iterations of algorithm; so SA does not accord with the physical annealing process of the solid material. In addition, it is obvious that the CSA has a quick convergence rather than SA.

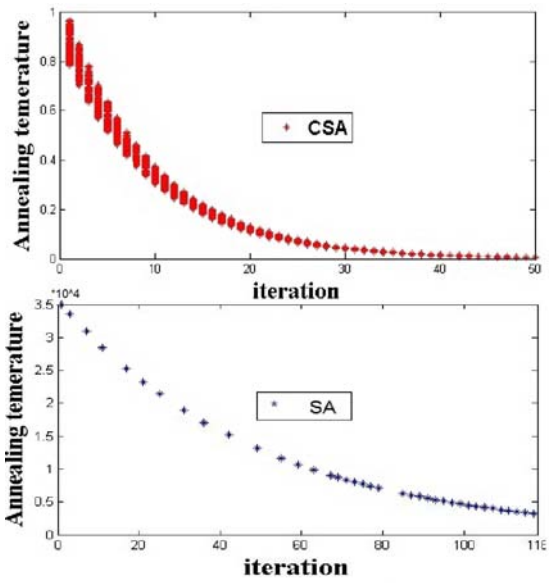

Fig. 4. Comparison of CSA backfire and re-annealing temperature process and SA temperature process, $\lambda = 0.9$.

4.3.2. Inversion Method to Local Search Phase

The local search used in this paper is the GA's inversion operator. This operator helps the algorithm to escape from local minima and find new neighbour solution if there is better solution in the neighbourhood. Inversion is an operator that generates off springs from a single parent. It first chooses two random cut points in a parent. The whole elements between the cut points are reversed. An example of the inversion operator is depicted below:

Before inversion: A B|B B A C|C C C

After inversion: A B| B C C A|B C C

The proposed search procedure goes on, once iteration of an inversion operator was done. If a local search is not improved, then it will move to the next iteration of inversion. Maximum iteration is five times.

5. Computational Results

In this paper, we proposed a CSA algorithm to solve MMALSP. Previous researches showed that evolutionary algorithms especially simulated annealing is an appropriate method for solving this problem; so, to validate CSA algorithm, we compared this algorithm with SA. We designed two groups of problem sets: small size and large size ones.

5.1. Small Size Problems

A set of ten problems is designed for small size problems where five problems have five workstations and models and five problems have eight workstations and ten models. The proposed CSA algorithm is applied to the above problems and its performance is compared, based on some comparison metrics, with the SA algorithm. Tables 4 and 5 present input values comprising operations time, stations length and MPS for small size problems, where ten problems are designed for this subject. The results show that CSA is superior to SA in terms of convergence speed (see figure 4). Figure 5 shows the comparison of computational time between SA and CSA. In addition, for the study of CSA performance, we compared this algorithm with SA using the relative percentage deviation (RPD) with the following formula:

$$RPD_i = \frac{A\lg_{sol}(i) - \min_{sol}(i)}{\min_{sol}(i)} \quad i = 1,...,n_{ps} \qquad (10)$$

$$\overline{RPD} = \frac{\sum_{i=1}^{n_{ps}} RPD}{n_{ps}} \quad i = 1,...,n_{ps} \qquad (11)$$

Where $A\lg_{sol}$ objective function's value for a given algorithm is, $\min_{sol}$ is the best value of objective function between both algorithms and n_{ps} is number of small size or large size problems. Figure 6 clearly shows that the CSA mean and lower values of its $\overline{RPD}$ are preferable to SA.

Table 4
Operation times on workstations and workstations length In small size problems

Product	Workstations							
	S1	S2	S3	S4	S5	S6	S7	S8
Product1	25	5	21	15	23	11	3	18
Product2	27	13	2	14	8	7	3	9
Product3	5	28	9	20	16	8	16	20
Product4	27	24	2	22	21	19	24	21
Product5	19	29	4	23	27	15	28	23
Product6	4	20	25	9	29	11	5	14
Product7	9	2	21	21	17	25	17	3
Product8	17	26	10	20	5	18	15	8
Product9	29	28	29	6	5	17	1	27
Product10	29	21	2	4	8	28	11	5
Stations length	10	11	13	10	12	6	12	11

Table 5
Small size problem set

problems	**a**(models)	MPS
PS1	5	(9,7,2,2,1)
PS2	5	(5,5,3,3,3)
PS3	5	(4,4,4,4,4)
PS4	5	(6,6,5,3,2)
PS5	5	(6,6,6,5,5)
PS6	10	(11,5,3,3,2,2,1,1,1,1)
PS7	10	(10,4,4,3,3,3,2,1,1,1)
PS8	10	(6,5,5,4,4,3,3,3,2,2)
PS9	10	(7,6,6,5,5,4,4,3,3,2)
PS10	10	(8,7,7,6,6,6,5,4,4,3)

Table 6
Large size problem set

problems	**a**(models)	MPS
PL1	10	(35,35,10,5,5,1,1,1,1,1,1,1,1,1,1)
PL2	10	(30 ,30,15,10,5,5,1,1,1,1 ,1, 1,1,1,1)
PL3	10	(20,20,15,15,10,6,6,6,1,1,1,1,1,1,1)
PL4	10	(15,15,15,10,10,10,10,5,5,4,4,3,1,1,1)
PL5	10	(8,8,8,8,8,8,7,7,7,7,7,7,7,6,6)

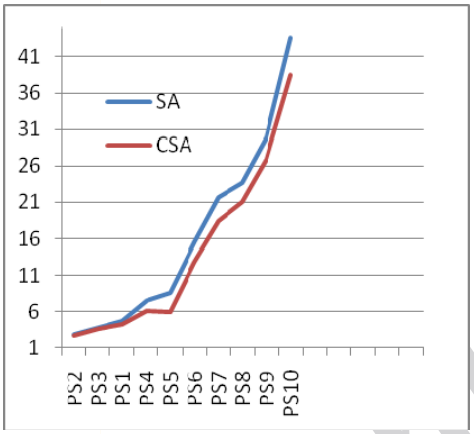

Fig. 5. The average of computational time for CSA and SA

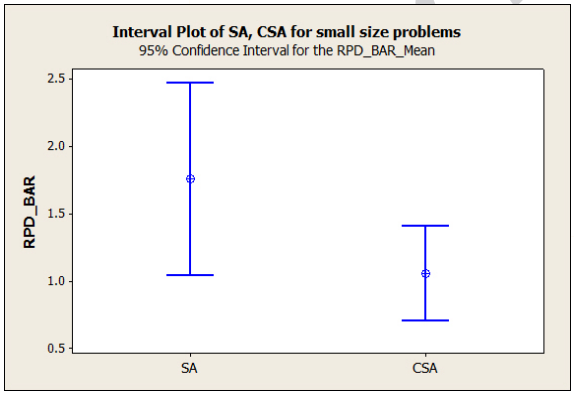

Fig. 6. The RPD mean for CSA and SA

Table 7
Bill of Assembly Time configuration

Product	WORK stations									
	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
Product1	25	5	21	15	23	11	3	18	4	5
Product2	27	13	2	14	8	7	3	9	29	26
Product3	5	28	9	20	16	8	16	20	1	18
Product4	27	24	2	22	21	19	24	21	23	17
Product5	19	29	4	23	27	15	28	23	25	5
Product6	4	20	25	9	29	11	5	14	26	26
Product7	9	2	21	21	17	25	17	3	3	19
Product8	17	26	10	20	5	18	15	8	13	11
Product9	29	28	29	6	5	17	1	27	9	16
Product10	29	21	2	4	8	28	11	5	24	13
Product11	6	23	14	15	25	9	6	25	14	3
Product12	29	23	12	29	8	23	24	17	27	8
Product13	29	12	23	11	25	23	10	30	6	5
Product14	15	20	24	18	8	12	16	3	9	6
Product15	24	6	6	7	28	17	6	14	5	8
Stations length	10	11	13	10	12	6	12	11	8	14

5.2 Large Size Problems

Since the proposed CSA algorithm must be efficient considering the MMALSP of real-world sized problems, another experiment is implemented for large sized problems; where five problems are designed with ten workstations and fifteen models. Table 6 presents the designed MPS for these five problems. Table 7 shows workstations length and assembly time configuration for total products in different workstations.

To compare the two algorithms, three metrics are considered. The first one is minimum objective function's value. Result of comparison is shown in figure 7. We can see the objective function value for CSA will significantly decrease rather than SA as the problem size increases. The average of computational time when run ten times for each problem is shown in figure 8. The results show that with increase problem size between two algorithms, the computational time of CSA for reaching a solution is much less than SA in large size problems. As Figure 9 illustrates, there is a significant difference between CSA and SA algorithms in term of the average of RPD measure at 95 percent confidence interval.

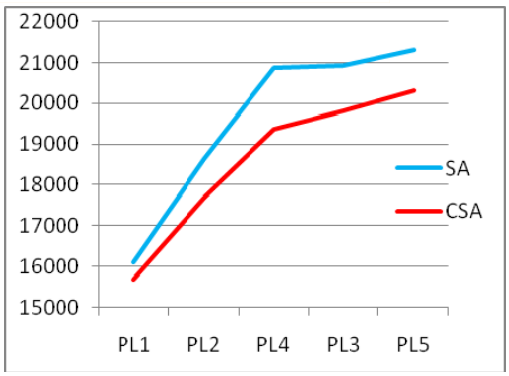

Fig. 7. Minimum OFV for CSA and SA

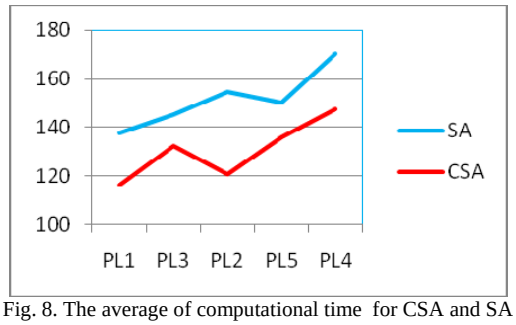

Fig. 8. The average of computational time for CSA and SA

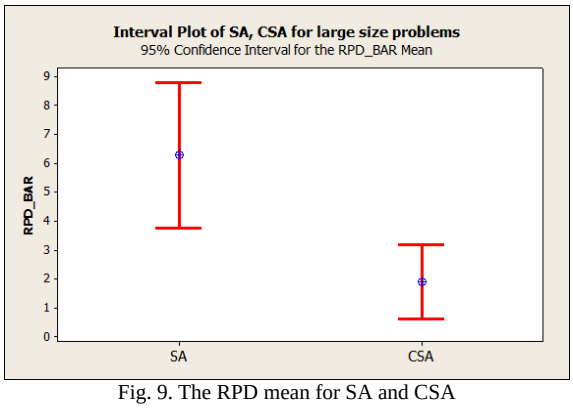

Fig. 9. The RPD mean for SA and CSA

6. Conclusions

In this study, a mixed-model assembly line sequencing problem was examined in JIT production systems. We considered minimizing line stoppage cost objective function as an important goal in MMALSP. Mathematical formulations for these objectives were provided and a new CSA approach was introduced to solve this problem. In order to make a valid comparison against competing solution methods, extensive two groups of test problems were provided and the reliability of the proposed CSA, based on some comparison metrics, was compared with SA algorithm as an appropriate approach to considering this problem. The results confirmed that the proposed CSA outperforms the SA approach in CPU time, objective function, and the average of RPD, especially in large size

7. References

[1] J. F. Bard, A. Shtub, S. B. Joshi, Sequencing mixed-model assembly lines to level parts usage and minimize line length. Int. J. Prod. Res., 32, 2431–2454, 1994.
[2] E. M. Dar-El, R. F. Cother, Assembly line sequencing for model mix. Int. J. Prod. Res., 13, 463–477, 1975.
[3] E. M. Dar-El, S. Cucuy, Optimal mixed-model sequencing for balanced assembly lines. Omega, 5, 333–342, 1977.
[4] L. Deyi, M. Haijun, S. Xuemei, Membership clouds and membership cloud generators. Journal of Computer Research and Development, 32 (6), 15–20, 1995.
[5] L. Deyi, D. Yi, Artificial intelligence with uncertainty. Chapman & Hall, 376, 2005.
[6] K. Di, L. Deyi, L. Deren, Cloud theory and its applications in spatial data mining knowledge discovery. Journal of Image and Graphics, 4A (11), 930–935, 1999.
[7] R. W. Eglese, Simulated annealing: A tool for operational research. European Journal of Operational Research, 46, 271–281, 1990.
[8] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Publishing Company, Inc., Reading, MA, 1989.
[9] C. J. Hyun, Y. Kim, Y. K. Kim, A genetic algorithm for multiple objective sequencing problems in mixed model assembly lines. Computers and Operations Research, 25, 7–8, 675–690, 1998.
[10] M. Kilbridge, L. Wester, The assembly line model-mix sequencing problem. In proceeding of the third international conference on operations research, Oslo (Paris: English Universities Press, Dunod Editeur), 247, 1963.
[11] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, Optimization by simulated annealing. Science, 220, 671-679, 1983.
[12] T. Korkmazel, S. Meral. Bicriteria, Sequencing methods for the mixed model assembly line in just-in-time production systems. European Journal of Operational Research, 131, 188–207, 2001.
[13] W. Kubiak, S. P. Sethi, A note on level schedules for mixed-model assembly lines in just-in-time production systems. Manag. Sci., 37, 121–122, 1991.
[14] V. Laarhoven, E. Aarts, Simulated annealing: Theory and applications. Kluwer Academic, Norwell, 186, 1987.
[15] S. A. Mansouri A multi-objective genetic algorithm for mixed-model sequencing on JIT assembly lines. European Journal of Operational Research, 167 (3), 696–716, 2005.
[16] P. R. McMullen, JIT sequencing for mixed-model assembly lines with setups using tabu search. Production Planning and Control, 9 (5), 504–510, 1998.
[17] P. R. McMullen, G. V. Frazier, A simulated annealing approach to mixed-model sequencing with multiple objectives on a JIT line. IIE Transactions, 32 (8), 679–686, 2000.
[18] P. R. McMullen, An efficient frontier approach to addressing JIT sequencing problems with setups via search heuristics. Computers and Industrial Engineering, 41, 335–353, 2001.
[19] P. R. McMullen, A Kohonen self-organizing map approach to addressing a multiple objective, mixed-model JIT sequencing problem. International Journal of Production Economics, 72, 59–71, 2001.
[20] P. R. McMullen, An ant colony optimization approach to addressing a JIT sequencing problem with multiple objectives. Artificial Intelligence in Engineering, 15, 309–317, 2001.
[21] P. R. McMullen, P. Tarasewich, A beam search heuristic method for mixed-model scheduling with setups. Int. J. Prod. Econ., 96, 273–383, 2005.
[22] N. Metropolis, A. Rosenbluth, M. Rosenbluth, E. Teller, Equation of state calculations by fast computing machines. Journal of Chemical Physics 21, 1087–1092, 1953.
[23] J. Miltenburg, G. Sinnamon, Scheduling mixed-model multi-level just-in-time production systems. Int. J. Prod. Res., 27, 1487–1509, 1989.
[24] J. Miltenburg, G. Steiner, S. Yeomans, A dynamic programming algorithm for scheduling mixed-model just-in-time production systems. Mathematical Computation Modeling 13, 57–66, 1990.

[25] Y. Monden, Toyota production system. Atlanta, GA: Institute if industrial engineers Press, 1983.
[26] K. Okamura, H. Yamashina, A heuristic algorithm for the assembly line model-mix sequencing problem to minimize the risk of stopping the Conveyor. International journal of production research, 17, 233-247, 1979.
[27] L. Pin,Y. Lin, Z. Jinfang, Cloud theory-based simulated annealing algorithm and application. Eng Appl Artif Intell, 22(4–5), 742–9, 2009.
[28] R. Tavakkoli-Moghaddam, A. R. Rahimi-Vahed, Multi-criteria sequencing problem for a mixed-model assembly line in a JIT production system. Applied Mathematics and Computation, 181 (2), 1471–1481, 2006.
[29] A. R. Rahimi-Vahed, M. Rabbani, R. Tavakkoli Moghaddam, S. A. Torabi, F. Jolai, A multi-objective scatter search for a mixed-model assembly line sequencing problem. Advanced Engineering Informatics, 21, 85–99, 2007.
[30] N. T. Thomopoulos, Mixed-Model line balancing with smooth station assignments. Management science, 16(9), 593-603, 1967.