Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 51-32 (2018) 600–605

A Fuzzy generalized simulated annealing for a simple assembly line balancing problem

M. Lalaoui*, A. El Afia**

**National School of Computer Science and Systems Analysis, Mohammed V University, Morocco (e-mail: med.lalaoui@yahoo.com)*

***National School of Computer Science and Systems Analysis, Mohammed V University, Morocco (e-mail: a.elafia@um5s.net.ma)*

Abstract: The assembly line is generally known as the last stage of the production processes. It constitutes the main production paradigm of the manufacturing industry. Thus, the performance of the assembly line problem has an important impact on the global performance of the entire production system. Among others, due to demand rate fluctuation. It's important to quickly rebalance the assembly line and obtain an effective solution for ALB problem. For these reasons, this article proposes an adaptive generalized simulated annealing using fuzzy inference system to solve simple assembly line balancing problem of type I (SALBP-I). The objective of the problem is to minimize the number of stations for a predefined cycle time of workstations in an existing assembly line. Moreover, the performance of our approach is analyzed using a well-known data set of the SALBP-I.

Keywords: Manufacturing, Adaptive control, Numerical simulation.

1. INTRODUCTION

A permanent challenge is faced by manufacturers to minimize the manufacturing cost while increasing the quality and accelerating the time to market to ensure that their products remain competitive as confirmed by Padron and al. 2009. The assembly costs represent more than 20% of the total manufacturing cost and more than 50% of the whole production time (Pan, 2005). Thus, the assembly line is a central issue that can alter the efficiency of all the manufacturing process. The resolution of these problems allows a significant increase in productivity, reducing in-process inventory, and removing bottlenecks.

The assembly line balancing problem (ALBP) aims at grouping the assembly operations and assigning them to stations, this with respect to the precedence constraints, so that the total assembly time at each station is equal and the selected performance measures are optimized (see Gu and al, 2007).

The ALB problems are classified as NP-hard problem thus an optimal line balance could not be found using an exact solution methods in reasonable times.

Hence, we resort to metaheuristics in order to propose a practical solution approach. Among several metaheuristic methods, we select the generalized simulated annealing (GSA) algorithm to study the effectiveness of the fuzzy controller on metaheuristic due to the following reasons. The first one is that the SA algorithm runs by using a single solution at a time, hence it does not cause memory shortage problems for even very large problem instances. The second reason is that the SA algorithm produces a feasible neighboring solution and do not need a repair algorithm which may lead to highly diversified solutions and deteriorates intensification.

To solve the ALBP, we intend to develop a controlled generalized simulated annealing based on fuzzy inference system that adjusts the parameters of the algorithm according to the knowledge obtained from its past search. The proposed algorithm is versatile enough to solve many types of ALBP.

In addition the performance of the proposed algorithm is assessed using a new simple assembly line balancing dataset released by Otto and Scholl, 2013.

The rest of this paper is structured as follows. Section 2 presents a detailed description of the generalized simulated annealing. Then, section 3 provides a description of the adaptive fuzzy generalized simulated annealing. The simple assembly line problem is formulated in section 4. Computational results and comparisons are presented in section 5. Finally, section 6 contains the conclusion of the study.

2. THE GENERALIZED SIMULATED ANNEALING ALGORITHM

The Simulated Annealing algorithm (SA) was first introduced by Kirkpatrick et al., 1983. The generalized Simulated Annealing (GSA) represents an improved version of the classical (SA). The GSA was proposed by (Tsallis and Stariolo, 1996) and it is based on Tsallis statistics (see Curado and Tsallis, 1992). This algorithm has been used in many classes of problems, such as chemistry and physics (Mundim et al., 2001; Zhaoxian and Dang, 2003), in the molecular systems optimization and the protein folding problems (Moretet al., 2002; Mundim and Tsallis, 1996).

The GSA algorithm covers a particular cases of the classical simulated annealing algorithm (Kirkpatrik et al., 1983) and the fast simulated annealing (Szu and Hartley, 1987). The GSA generalizes the distribution functions of the neighborhood, the probability of acceptance, and the temperature procedure of the CSA and FSA. Therefore, instead of using the Gaussian or the Cauchy–Lorentz neighborhood distribution function, the GSA uses a function defined as,

$$g_{q_v}(x) = \left\{\left(\frac{q_v-1}{\pi}\right)^{D/2}\left[\Gamma\left(\frac{1}{q_v-1}+\frac{D-1}{2}\right)\Big/\Gamma\left(\frac{1}{q_v-1}-\frac{1}{2}\right)\right]\left[T_{q_v}(t)\right]^{-\left(\frac{D}{3}-q_v\right)}\right\} \times \frac{1}{\left\{1+(q_v-1)x^2/\left[T_{q_v}(t)\right]^{2/3-q_v}\right\}^{\frac{1}{q_v}-1+D-\frac{1}{2}}} \quad (1)$$

Which has the following limits,

$$\lim_{q_v\to 1} g_{q_v}(x) = g^{CSA}(x) = \frac{e^{x^2/T_{q_v}(t)}}{[\pi T_1(t)]^{D/2}} \quad (2)$$

$$\lim_{q_v\to 2} g_{q_v}(x) = g^{FSA}(x) = \frac{\Gamma\left(\frac{D+1}{2}\right)}{\pi^{D+1/2}}\cdot\frac{T_2(t)}{[T_2(t)^2+x^2]^{D+1/2}} \quad (3)$$

These limits are the neighbourhood distribution functions of the CSA (equation 2) and the FSA (equation 3).

The GSA function of probability of acceptance is defined as,

$$A_{acc=}\left[1+(q_a-1)\frac{E(C_{t+1})-E(C_t)}{T_{q_v(t)}}\right]^{-1/q_a-1} \quad (4)$$

10.1016/j.ifacol.2018.11.489

Which has the following limits when $q_a \to 1$

$$\lim_{q_a \to 1} A_{acc} = e^{[E(C_t)-E(C_{t+1})]/T_{q_v}(t)} \quad (5)$$

That is the acceptance probability function of the CSA (Kirkpatrik et al., 1983) and FSA algorithms (Szu and Hartley, 1987), where t represents the artificial time and D is the problem size. This visiting distribution is used to generate a trial jump distance $\Delta x(t)$ of variable $x(t)$ under artificial temperature $T_{q_v}(t)$. The trial jump is accepted if the objective function is improved, otherwise it might be accepted according to an acceptance probability. The generalized Metropolis algorithm is used for the acceptance probability:

$$p_{q_a} = \min\left\{1, [1-(1-q_a)\Delta f]^{\frac{1}{1-q_a}}\right\}, \quad (6)$$

Where q_a is the acceptance parameter $q_a \in]1,3[$. For $q_a < 1$, a zero acceptance probability is assigned to the cases where,

$$[1-(1-q_a)\beta\Delta f] < 0 \quad (7)$$

The GSA temperature function is defined as,

$$T_{q_v}(t) = T_0 \frac{2^{q_v-1}-1}{(1+t)^{q_v-1}-1} \quad (8)$$

Which,

$$\lim_{q_v \to 1} T_{q_v}(t) = T_1(t) = T_0 \frac{\ln 2}{\ln(1+t)} \quad (9)$$

$$\lim_{q_v \to 2} T_{q_v}(t) = T_2(t) = \frac{T_0}{t} \quad (10)$$

These limits are the temperature functions of the CSA and FSA described by equation (9) and (10) respectively.
The behavior of GSA depends on many uncertain factors. One of the important factors is a balance between exploitation and exploration of the search space. To overcome this, researchers apply many approach to deal with this difficulty as in Lalaoui et al. (2016, 2017, 2018a, 2018b), Aoun et al. (2016, 2018), Bouzbita et al. (2018a, 2018b), El Afia et al. (2017a, 2017b) and Kabbaj et al. (2016, 2018).

This balance between exploitation and exploration is strongly impacted by the design strategy of the GSA parameters such as the neighbourhood parameter q_v and the acceptance parameter q_a.
These parameters are controlled by the fuzzy logic controlled and automatically regulated during the search process. Thus, much effort for the fine-tuning of these parameters can be gained and the GSA search ability in finding the global optimum can be enhanced.

3. THE FUZZY GENERALIZED SIMULATED ANNEALING

3.1. Fuzzy logic controller (FLC)

The fuzzy logic controller (FLC) is based on the fuzzy theory, it was proposed by (Mandani, 1974). The FLC allows the input–output relations of a system to be set and reflects a knowledge that cannot be represented quantitatively such as expert's views. The structure of the fuzzy logic controller is shown in figure 1.The Inference system is based on knowledge base (rules) that infers the output according to the input that comes from the controlled system. The fuzzifier and the defuzzifier transform crisp numbers into fuzzy number and the fuzzy numbers into crisp numbers respectively.

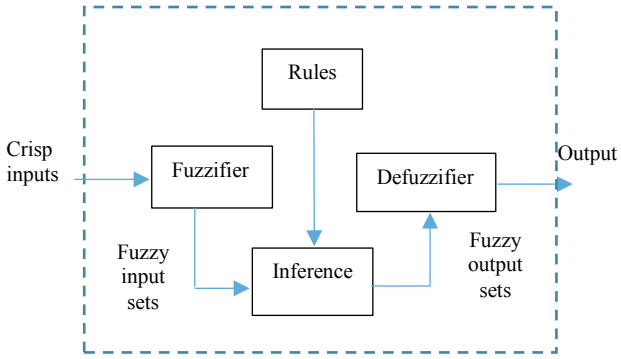


Figure 1. The structure of the fuzzy logic controller

3.2. The Design of the FLC

The main idea of our approach is to enhance the performance of the generalized simulated annealing through the dynamic variation of the q_a and q_v parameters. These parameters have a huge effect on the performance of GSA. In addition, it appears impossible to fix a universally best q_v, and q_a because they highly depend on the instance of the problem to optimize. It was reported by Flavia et al., 2006 that these parameters have a better interval of values ranging from 1.1 to 1.9 for q_v , and 1.1 to 2.9 for q_a . The GSA recovers the classical SA when $q_v = q_a = 1$ and it recovers the special case of FSA when $q_v = 2$ and $q_a = 1$. Also, the cooling is faster than that of the CSA and the FSA when $q_v > 2$. A bigger q_v (> 3) makes the cooling faster and a negative q_a with bigger absolute value makes GSA accepts less hill climbing. All these empirical experiments done in previous studies were integrated into the FLC's knowledge base.
That is, the proposed FLC considers the average of the cost function generated by the GSA. Then, depending on the values obtained we adjust the neighbourhood parameter q_v and the acceptance parameter q_a for the next generation.

To automatically adjust these parameters, we use the approach introduced by Wang, Wang and Hu (1997). The neighborhood FLC and the acceptation FLC are implemented independently to automatically adjust the neighbourhood parameter and the acceptance parameter during the search process. The controlling strategy for the neighbourhood parameter and the acceptance parameter is to consider the evolution of the average cost function of two continuous generations $\Delta f(V\,;\,t-1)$ and $\Delta f(V\,;\,t)$.
Based on many experimental data and domain expert opinion, we adopted the fuzzy decision table as presented in table 1. Then, the input values are normalized into integer values in the range $[-4.0, 4.0]$. The scaled inputs are assigned to the corresponding indexes i and j in the fuzzy decision table as illustrated in table 2. The membership functions of these fuzzy linguistic variables are shown in figure 2.

Table 1. Fuzzy decision table

		$\Delta f(t-1)$								
		NR	NL	NM	NS	ZE	PS	PM	PL	PR
$\Delta f(t)$	NR	NR	NL	NL	NM	NM	NS	NS	ZE	ZE
	NL	NL	NL	NM	NM	NS	NS	ZE	ZE	PS
	NM	NL	NM	NM	NS	NS	ZE	ZE	PS	PS
	NS	NM	NM	NS	NS	ZE	ZE	PS	PS	PM
	ZE	NM	NS	NS	ZE	ZE	PS	PS	PM	PM
	PS	NS	NS	ZE	ZE	PS	PS	PM	PM	PL
	PM	NS	ZE	ZE	PS	PS	PM	PM	PL	PL
	PL	ZE	ZE	PS	PS	PM	PM	PL	PL	PR
	PR	ZE	PS	PS	PM	PM	PL	PL	PR	PR

NR – Negative larger, NL – Negative large,
NM – Negative medium, NS – Negative small,
ZE – Zero, PS – Positive small,
PM – Positive medium, PL –Positive large,
PR – Positive larger,

Table 2. Values for Fuzzy logic controller

$z(i,j)$		i								
		-4	-3	-2	-1	0	1	2	3	4
j	-4	-4	-3	-3	-2	-2	-1	-1	0	0
	-3	-3	-3	-2	-2	-1	-1	0	0	1
	-2	-3	-2	-2	-1	-1	0	0	1	1
	-1	-2	-2	-1	-1	0	0	1	1	2
	0	-2	-1	-1	0	0	1	1	2	2
	1	-1	-1	0	0	1	1	2	2	3
	2	-1	0	0	1	1	2	2	3	3
	3	0	0	1	1	2	2	3	3	4
	4	0	1	1	2	2	3	3	4	4

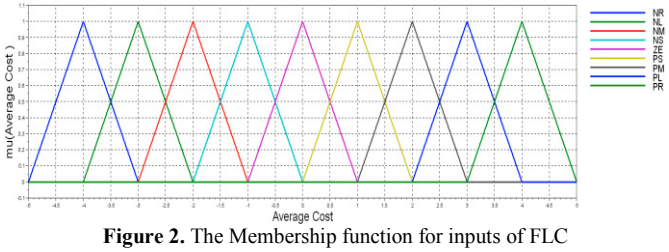


Figure 2. The Membership function for inputs of FLC

According to the result of the above process, the neighbourhood parameter and the acceptance parameter are computed as follows:

$$\Delta v(t) = \alpha \cdot z(i,j), \quad \Delta a(t) = \beta \cdot z(i,j) \quad (11)$$

Where α and β are a given values to adjust an increasing and decreasing range for the neighbourhood parameter and the acceptance parameter (e.g. $\alpha = 0.02, \beta = 0.002$).

The neighbourhood parameter and and the acceptance parameter are then calculated by the following equations:

$$q_v(t) = \Delta c(t) + q_c(t-1) \quad (12)$$

$$q_a(t) = \Delta a(t) + q_a(t-1) \quad (13)$$

Where $q_v(t)$ and $q_a(t)$ the neighbourhood parameter and the acceptance parameter at generation t, respectively.

3.3. GSA algorithm using FLC

In this study, we determine the cooling schedule of SA. The process of the proposed FGSA is shown in algorithm 1. First, the fuzzy logic controller is initialized, and the initial solution and temperature are set. Then, the Fuzzy Generalized Simulated Algorithm (FGSA) goes through similar processes as GSA. The only difference is that the temperature schedule and the acceptance probability is controlled using FLC. Where the next temperature is determined according to the current solution state and the actual temperature. The FGSA continuously observes the status of the current solution and then determine the next temperature and the next acceptance probability accordingly.

Algorithm 1 : The proposed fuzzy simulated annealing

Input: a permutation of tasks obtained from the random-key methods(S^*);
Maximum number of neighbourhood search for each temperature stage (Max_Iter)
Output: the best solution for the balancing problem (S_{best})
Initialize fuzzy logic controller()
Initialize the algorithm's parameters (T_0, T_f, λ, PL)
Set $T_k = T_0$
Generate an initial solution (S_0) for the balancing problem
Set $S_c = S_0, S^* = S_0$
Compute the value of the objective for the initial solution $f(S_0)$
Set $f(S_c) = f(S_0), f(S^*) = f(S_0)$
While $(T_k > T_f)$
  Set k=1
  While (k<= Max_Iter)
    Generate a neighbour solution (S_N) for the problem by consideration the priority of the current solution S_c
    Compute objective function of neighbour solution ($f(S_N)$)
    Set $\Delta f = f(S_N) - f(S_c)$
    If $(\min\left\{1, [1-(1-q_a)\Delta f]^{\frac{1}{1-q_a}}\right\} > random[0,1]$ or $\Delta f < 0$
      Accept S_N as the current solution;
      Set $k = k + 1$
    End If
    Update the best solution (S_{best})
    Update q_v using fuzzy logic controller(S^*, T)
    Update q_a using fuzzy logic controller(S^*, T)
    Set $T(k) = T_0 \frac{2^{q_v-1}-1}{(1+k)^{q_v-1}-1}$
  End
End

According to the result of the above process, the neighbourhood parameter and the acceptance parameter are computed as follows:

$$\Delta v(t) = \alpha \cdot z(i,j), \quad \Delta a(t) = \beta \cdot z(i,j) \quad (11)$$

Where α and β are a given values to adjust an increasing and decreasing range for the neighbourhood parameter and the acceptance parameter (e.g. $\alpha = 0.02, \beta = 0.002$).

The neighbourhood parameter and and the acceptance parameter are then calculated by the following equations:

$$q_v(t) = \Delta c(t) + q_c(t-1) \quad (12)$$

$$q_a(t) = \Delta a(t) + q_a(t-1) \quad (13)$$

Where $q_v(t)$ and $q_a(t)$ the neighbourhood parameter and the acceptance parameter at generation t, respectively.

3.3. GSA algorithm using FLC

In this study, we determine the cooling schedule of SA. The process of the proposed FGSA is shown in algorithm 1. First, the fuzzy logic controller is initialized, and the initial solution and temperature are set. Then, the Fuzzy Generalized Simulated Algorithm (FGSA) goes through similar processes as GSA. The only difference is that the temperature schedule and the acceptance probability is controlled using FLC. Where the next temperature is determined according to the current solution state and the actual temperature. The FGSA continuously observes the status of the current solution and then determine the next temperature and the next acceptance probability accordingly.

In the rest of this work is intended to solve SALBP-1 and to verify the performance of the FGSA.

4. SOLVING THE SALBP-1 WITH FGSA

4.1. Mathematical model of SALBP-1

The notations that will be used throughout the paper are described as follows:

C: the cycle time.
i: the task identifier, where $i \in N$
k: the workstation identifier, where $k \in M$
n: the number of tasks.
m : the number of workstations in the assembly line.
M: the set of workstations, where $M = \{1,2,..,m\}$
N: the set of tasks.
P': the precedence matrix, $P'(i,j) = 1$, if task i is an immediate or indirect predecessor of task j;$P'(i,j) = 0$, otherwise.
$s(k)$: the set of tasks which are currently assigned to workstation k.
SI: the smooth index, where :

$$SI = \sqrt{\frac{1}{m-1}\sum_{k=1}^{m}(C - ST_k)^2} \quad (14)$$

ST_k : the processing time of the kth workstation,

$$ST_k = \sum_{i \in s(k)} t_i \quad (15)$$

t_i : the processing time of task i
x_{ik}: the binary variable. $x_{ik} = 1$, if task i is assigned to workstation k; $x_{ik} = 0$; otherwise.

The objective function of SALBP-1 is to minimize the number of the workstations, i.e.

$$minimise \left(m \times \sqrt{\sum_{k=1}^{m} (c - t_k)^2} \right) \quad (16)$$

Where m is the number of stations, c is the cycle time and t_k is the workload at station k. This objective function simultaneously improves the number of workstations and the smoothness of the line balancing and also modifies the search space allowing more variation in cost values. The Smoothness measures the uniformity of the tasks distribution along workstations (see Scholl, 1999).

Contraints conditions :

$$\sum_{k \in M} x_{ik} = 1, \forall i \quad (17)$$

$$\sum_{k \in M} k \cdot x_{ik} \leq \sum_{k \in M} k \cdot x_{jk}, \quad \forall P'(i,j) = 1 \quad (18)$$

$$\sum_{i \in N} x_{ik} \cdot t_i \leq C, \quad \forall k \quad (19)$$

$$x_{ik} \in \{0,1\}, \quad \forall i,k \quad (20)$$

Equation (17) ensures that every task is assigned to one and only one workstation. Constraint (18) defines the precedence relationships between tasks i and j. Constraint (19) ensures that the processing time of every workstation has not exceeded the cycle time. Finally, (20) indicates that all x_{ij} variables in the model are binary variables.

4.2. The encoding scheme

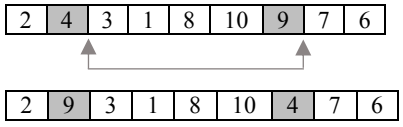
Due to the continuous nature of the neighbourhood function of the GSA algorithm, the standard scheme of the GSA cannot be applied directly to SALB-1. So an appropriate mapping is required to represent a permutation through floating-point vectors (i.e. neighbourhood solution). These issues were tackled according to procedures frequently applied in the literature (see Hamta et al. 2013) and are summarized as:

- *Solution representation*

The Task Oriented representation was chosen, which means that a solution is represented as an array with a size equal to the number of assembly tasks. Nearchou et al, 2011 found in a preliminary experiments that the Task Oriented (TO) representation was adequate with respect to solution quality of the SALBP. Due to the likeness between Nearchou's works and our case, we decided to use TO representation. Each array contains a task index. However, in the GSA search procedure, a candidate solution is represented by an array of real numbers. Hence the solution in this case uses the random keys procedure (see Hamta et al. 2013) to map a candidate solution array into a tasks indexes array. This procedure maps the smallest value in the position array of a candidate solution into the number 1, the second smallest value into the number 2, and so on. This is repeated until all array values are mapped into a task index;

After the random-key method is applied and a permutation of integers is produced, one or more neighbourhood operations is performed on it. Thus, the neighbourhood $N(S)$, is characterized as follows, one for each l:

- Select two tasks i_1 and i_2 $(i_1 \neq i_2)$ randomly.
- Swap tasks i_1 and i_2



- Constraints treatment

As mentioned before, two different types of constraints have to be considered in order to solve SALBP-1 instances. First of them is the precedence constraint. We applied the procedure described by Hamta et al. 2013. Given a task indexes array, and following a precedence graph, the procedure corrects the sequence to be arranged in a way that no successor precedes a predecessor in the adjusted array. After that, capacity constraint should be dealt with. To do so, we use the following procedure: a workstation is opened and then the maximum number of tasks in the array with total time smaller than cycle time is assigned to the opened workstation. Then this workstation is closed and a new one is then opened. This procedure is repeated until all tasks are assigned to a workstation. A detailed explanation of capacity constraint treatment in Task-Oriented approach is provided by Scholl, 1997.

- *Repairing infeasible solutions*

An infeasible solution can be generated by the random-keys procedure. So, an appropriate method is needed to repair the infeasible solutions. For this reason, we implemented in this paper a methods identical to those used by Nearchou, 2007 with a few changes. The precedence relationships are defined in matrix M in which $M_{ij} = 1$ if task i must be ended immediately before task j, otherwise $M_{ij} = 0$. The procedure to construct the precedence matrix M is given by the algorithm 2.

Algorithm 2: Create the precedence matrix M

Input : n the number of tasks, M the precedence matrix
Output : M the precedence matrix
For $i = 1$ **to** number of tasks **do**
  For $j = 1$ **to** number of tasks **do**
    If task i must be completed before task j **then**
      $M_{ij} = 1$ **else** $M_{ij} = 0$
    End if
  End for
End for
For $i = 1$ **to** Number of tasks **do**
  For $j = 1$ to Number of tasks **do**
    If $M_{ij} = 1$ **then**
      for $k = 1$ **to** Number of tasks **do**
        $M_{kj} = M_{kj} + M_{ki}$
        If $M_{kj} = 2$ **then**
          $M_{kj} = 1$
        End if
      End for
    End if
  End for
  For $j = 1$ **to** number of tasks **do**
    count=0
    For i=1 to number of tasks **do**

```
            If M_ij = 1 then
              count=count+1
            End If
        End for
        M_{n+1,j} = count // where n is the of tasks
End For
Return M
```

The indirect predecessors of each task are represented in algorithm 2. When task k is an indirect predecessor of task j the procedure sets $M_{kj} = 1$. In addition, the total number of the predecessors of every task is counted, and is conserved in the column of the $(n+1)th$ row of matrix M. This information is then utilized in the repairing procedure presented by algorithm 3.

Algorithm 3: Repair the infeasible solution

Input: an infeasible solution S
Output : the repaired and feasible solution corresponding to FS
$M_p = M$ *// make a copy of M and work with it*
$J = 0$ *//index of array PS (partial schedule*
$i = 1$
While $J < n$ **do** *//n is the number of tasks*
$a = S(i)$
If ($M_{p_{n+1,a}} = 0$) **and** (a is not in array PS) **then**
$J = J + 1$
$PS[J]$=a
For $k = 1$ **to** n **do** // Update Mp
If $Mp_{a,k} = 1$ **then**
$Mp_{a,k} = 0$
$Mp_{n+1,k} = Mp_{n+1,k} - 1$
End if
End for
End If
$i = i + 1$
If $i > n$ **then** $i = 1$ **End if**
End while
Return *FS*

The algorithm 3 corrects the infeasible solution S and returns the feasible version. Firstly, FS is empty and iteratively, each feasible task a is inserted in the next existing position to create a string of tasks such that precedence relations are met.

Then the procedure employed to solve the SALBP-1 consists in running the FGSA (algorithm 1) and all mapping, correction and assignment occur when the evaluation of the cost function is required.
In the next section, experiments were designed to measure the effects of hybridization of FLC and GSA and to show how the proposed approach can improve the solution quality.

5. COMPUTATIONAL RESULTS

The GSA and the FGSA algorithms have been implemented in JULIA language and run on instances taken from Otto et al., 2013 using a single core of an Intel Core i7- 6500U 2.5 gigahertz processor, with 8 gigabytes of memory.
The basic characteristics of the problems are summarized in table 3. The first column qualifies the problem size (small, medium and large instances). The second column shows the number of tasks. The third column shows the trickiness level of each problem. The fourth column gives the optimum number of stations found with SALOMON. We have run SALBP-1 with both versions of GSA 30 times.

Table 3. The Experimental results

	Instance	Category	No of stations in optimum (SALOMON)	GSA	FGSA
Small	n=20_92	Tricky	11	11	11
	n=20_93	very tricky	13	13	13
	n=20_94	extremely tricky	10	10	10
	n=20_101	less tricky	13	**12**	**12**
Medium	n=50_26	extremely tricky	27	28	**27**
	n=50_138p10	Tricky	12	12	12
	n=50_51	less tricky	12	12	12
	n=50_54	very tricky	11	11	11
Large	n=100_34	less tricky	15	15	15
	n=100_35	tricky	15	16	**15**
	n=100_36	extremely tricky	14	15	**14**
	n=100_40	very tricky	14	15	**14**

As it can be observed in table 3 the GSA and the FGSA produce solutions that are as good as the SALOMON except for the extremely and the very tricky instance of the large problem. In addition, for the fourth instance of the small problem the GSA and FGSA give a better solution.
Note also that the fuzzy controller improves the solution given by the vanilla GSA especially for large instances. Obviously, the proposed FGSA can produce a better balancing effect than GSA.

In table 4, the proposed FGSA shows better results than the vanilla version of the GSA in term of the best cost function, the average cost function and also the average CPU time. This means that the FLC enable the GSA to search more efficiently.
Concerning the average CPU time, the FGSA is faster than the GSA, this demonstrates that the proposed FLC adjusted the neighbourhood parameter and the acceptance parameter of the GSA.
The significance of the results has been evaluated using the Wilcoxon signed-rank test, which is a non-parametric test, to compare the performance of GSA and FGSA. Table 4 provides the p-values and the corrected p-values with Benjamini–Hochberg correction. These values tell us, that the FGSA outperforms GSA at 5%-significance. Based on these computational results, we conclude that applying the FLC in GSA can be a good way to improve the performance and to find better solutions for the SALBP-1.

Table 4. Performance of the GSA compared to FGSA for large instances of SALBP-1

		n=100_ 34	**n=100_ 35**	**n=100_ 40**	**n=100_ 36**
GSA	Best cost function	-4273	-3148	-1415	-3115
	Average cost	-4956	-3421	-2934	-4024
	Average CPU Time (sec.)	2.4	3.1	3.9	4.7
FGSA	Best cost function	-4021	-2913	-1197	-3208
	Average cost	-3527	-3613	-2613	-3291
	Average CPU Time (sec.)	2.1	2.7	3.5	3.7
Wilcoxon signed-rank test	p-Value	0.00	0.01	0.03	0.07
	Corrected p-value	0.00	0.04	0.05	0.11

6. CONCLUSIONS

This paper main contribution is to evaluate the effectiveness of the fuzzy simulated annealing in order to solve the Simple Assembly Line Balancing Problem-type 1. The vanilla version of the generalized simulated annealing and the fuzzy controlled one have been employed and compared against results obtained using the exact procedure named SALOME.

The results obtained show that the GSA was effective and the fuzzy controller was able to enhance the algorithm for the proposed problem instances. As future work we propose to tackle other ALBP versions with GSA and fuzzy control. Moreover, this study could be extended by the comparison of GSA performance on solving SALBP-1 with very larger data sets and comparing its results with other metaheuristic approaches.

7. REFERENCES

Aoun, O., Sarhani, M., El Afia, A. (2016). Investigation of hidden markov model for the tuning of metaheuristics in airline scheduling problems. Proceedings of the 14th symposium on control in transportation systems, Istanbul, Turkey (pp. 347-352).

Aoun, O., El Afia, A. Salvador, G. (2018). Self inertia weight adaptation for the particle swarm optimization. Proceedings of the international conference on learning and optimization algorithms: theory and applications, Rabat, Morocco.

Bouzbita, S., El Afia, A., Faizi, R. (2018). Hidden Markov Model classifier for the adaptive ACS-TSP pheromone parameter. In bioinspired heuristics for optimization, pp.143-156, Springer.

Bouzbita, S., El Afia, A., Faizi, R. (2018). Parameter adaptation for ant colony system algorithm using hidden markov model for tsp problems. Proceedings of the international conference on learning and optimization algorithms: theory and applications, Rabat, Morocco.

Curado, EMF., Tsallis, C. (1991). Generalized statistical mechanics: connection with thermodynamics, Journal of Physics a: mathematical and general 24 (2), L69.

El Afia, A., Bouzbita, S., Faizi, R. (2017). The Effect of Updating the Local Pheromone on ACS Performance using Fuzzy Logic. International journal of electrical and computer engineering (IJECE), 7(4), 2161-2168.

El Afia, A., Kabbaj, MM. (2017). Supervised learning in Branch-and-cut strategies. Proceedings of the 2nd international conference on big data, cloud and applications, Tetouan, Morocco.

Flavia P., Agostini, D., (2006). A generalized simulated annealing applied to protein folding studies, Wiley Periodicals, Inc. Wiley InterSciene.

Gu L., Hennequin S, Sava A, Xie X. (2007). Assembly line balancing problems solved by estimation of distribution. Proceedings of the 3rd IEEE international conference on automation science and engineering, IEEE CASE, pp 123–127.

Kabbaj, M. M., El Afia, A. (2016). Towards learning integral strategy of branch and bound. Proceedings of the IEEE 5th international conference in multimedia computing and systems, Marrakech, Morocco, (pp. 621-626).

Kabbaj, M. M., El Afia, A. (2018). Model selection for learning Branch-and-cut strategies. Proceedings of the international conference on learning and optimization algorithms: theory and applications, Rabat, Morocco.

Kirkpatrik,S. (1983). Optimization by simulated annealing. Science, Vol. 220, No. 4598, 671–680.

Lalaoui, M., El Afia, A., Chiheb R. (2016). Hidden Markov Model for a self-learning of Simulated Annealing cooling law. Proceedings of the 5th International Conference on Multimedia Computing and Systems, Marrakech, Morocco, (pp. 558-563).

Lalaoui, M., El Afia, A., Chiheb R. (2017). A self-adaptive very fast simulated annealing based on Hidden Markov model. Proceedings of the 3rd international conference of cloud computing technologies and applications, Rabat, Morocco.

Lalaoui, M., El Afia, A., Chiheb R. (2018). A self-tuned simulated annealing algorithm using hidden markov model. International Journal of Electrical and Computer Engineering 8(1):291-298.

Lalaoui, M., El Afia, A., Chiheb R. (2018). Simulated annealing with adaptive neighborhood using fuzzy logic controller. Proceedings of the international conference on learning and optimization algorithms: theory and applications, Rabat, Morocco.

Lv, H., Lu C. (2010). An assembly sequence planning approach with a discrete particle swarm optimization algorithm. Int J Adv Manuf Technol 50(5–8):761

Mamdani,E. H. (1974). Application of fuzzy algorithms for simple dynamic plant. Proc. Inst. Elect. Eng., vol. 121, pp. 1585–1588.

Moret MA, Pascutti PG, Mundim KC, Bisch PM and Nogueira Jr E (2001). Multifractality, Levinthal paradox, and energy hypersurface. Phys Rev E 63:R1-4.

Mundim, KC., Tsallis C., (1996). Geometry optimization and conformational analysis through generalized simulated annealing. Int J Quantum Chem 58:373-381.

Mundim KC, Liubich V, Dorfman S, Felsteiner J, Fuks D and Borstel G (2001). Nonempirical simulations of boron interstitials in tungsten. Physica B 301:239-254.

N. Hamta, S. M. T. Fatemi Ghomi, F. Jolai, and M. Akbarpour Shirazi. (2013). A hybrid PSO algorithm for a multi-objective assembly line balancing problem with flexible operation times, sequence-dependent setup times and learning effect," International Journal of Production Economics, vol. 141, no. 1, pp. 99–111.

Nearchou, A.C., (2007). Balancing large assembly lines by a new heuristic based on differential evolution method. International Journal of Advanced Manufactur- ing Technology 34, 1016–1029.

Nearchou, A.C., (2011). Maximizing production rate and workload smoothing in assembly lines using particle swarm optimization. International Journal of Production Economics 129, 242–250.

Otto, A., Otto, C., & Scholl, A. (2013). Systematic data generation and test design for solution algorithms on the example of SALBPGen for assembly line balancing. European Journal of Operational Research, 228(1), 33–45.

Padron M, de los AIM, Resto P, Mejia HP (2009). A methodology for cost-oriented assembly line balancing problems. J Manuf Technol Manag 20(8):1147–1165

Pan C., (2005). Integrating CAD files and automatic assembly sequence planning. Ph.D. thesis, Iowa State University.

Scholl, A., (1999) Balancing and sequencing of assembly lines. Physica- Verlag Heidelberg,

Scholl, A., & Klein, R. (1997). SALOME: A Bidirectional Branch-and-Bound Procedure for Assembly Line Balancing, Informs Journal on Computing 9(4):319-334.

Szu,H.H. , R.L. Hartley, 1987. Fast simulated annealing. Phys. Lett. A 122, 157–162.

Tsallis, C., Stariolo, D.A., (1996). Generalized simulated annealing Physica A 233 395.

Wang, P. T., Wang, G. S. and Hu, Z. G. (1997) Speeding up the Search Process of Genetic algorithm by Fuzzy Logic, Proc. of the 5th European Congress on intelligent Techniques and Soft Computing, pp. 665-671.

Zhaoxian, Y., Dang M. (2003). Generalized simulated annealing algorithm applied in the ellipsometric inversion problem, Thin Solid Films 425:108-112.