# Multi-objective Assembly Line Balancing Using Fuzzy Inertia-adaptive Particle Swarm Algorithm

**Simona DINU**

Fundamental Sciences and Humanities Department,
Constanta Maritime University,
104, Mircea cel Batran Street, Constanta, code 900663, Romania,
simona.dinu@cmu-edu.eu

**Abstract:** The Assembly Line Balancing problem is an industrial optimization problem of considerable importance in lean systems. It has been extensively studied in literature through classical optimization methods. However, conventional computing paradigms have not proved practical utility for complex problems. Metaheuristic solutions such as "Tabu Search", "Simulated Annealing", "Genetic Algorithms", "Evolutionary Programming", "Ant Colony", "Particle Swarm Optimization" were a preoccupation mainly for the last two decades. This paper presents a model of a multi-objective Assembly Line Balancing problem and a solution approach based on Particle Swarm Optimization (PSO) with a fuzzy controller for tuning inertia weight. This prevents the premature convergence and, in addition, the algorithm demonstrates improved search features. For the considered test instance, the algorithm obtains a better result compared to the results reported in the literature, regarding the number of stations actually used, the line efficiency, the total unused time, the variation in charging stations and the uniformity index of the line.

**Keywords:** Particle Swarm Optimization (PSO), Assembly Line Balancing (ALB) problem, fuzzy controller, multi-objective optimization.

## 1. Introduction

As part of an industrial manufacturing system, installing an assembly line is a costly decision and requires a considerable time for execution and therefore it is important to be well designed and properly balanced to guarantee maximum efficiency in operation.

An important assembly design problem is the assembly line balancing (ALB) problem. This decisional problem is a classic Operations Research (OR) optimization problem that aims to determine the allocation of the tasks to an ordered sequence of workstations such that every task is assigned at just one station, the precedence relations are not violated and certain objectives are fulfilled.

Since the bin-packing problem, which is an ALB problem without precedence constraints [5], is NP-hard, even the simple case of the ALB problem is NP-hard by nature. Indeed, m tasks and r preference constraints generate m!/2r feasible solutions of the problem [2], as there are m!/2r possible task sequences. As one can observe, the problem size grows very rapidly with the number of tasks and/or workstations. Because of the high computational complexity, conventional optimization methods do not seem appropriate for this simple or multi-objective practical optimization problem.

Due to the complexity of the ALB problem and its practical importance for industrial applications, many approaches based on meta-heuristics such as Tabu Search, Simulated Annealing, Evolutionary Algorithms, Agent -based approaches (Ant Colony Optimization and Particle Swarm Optimization) or hybrid Artificial Intelligence methods have been applied recently in attempts to solve this manufacturing optimization problem. A survey study of soft computing applications in ALB problems is presented in [15]. Other comprehensive reviews of assembly systems and different balancing problems are presented in [2].

This study proposes a model and a solution approach to a multi-objective ALB problem considering three evaluation criteria. This multi-objective problem is solved by a discrete PSO algorithm whose efficiency is enhanced due to the development of a fuzzy controller for tuning inertia weight.

## 2. Assembly Line Balancing (ALB) Problems: Basic Concepts and Typologies

Assembly lines are production lines consisting of several consecutive workstations ($i=1,...,m$) located along a conveyor belt that transports the production units through the line with a constant transportation speed.

The total work necessary to achieve the final commodity is divided into n elementary operations, called tasks [23]. Each station executes successively more tasks ($j=1,...,n$); each task requires $t_j$ units of time for completion and certain equipment and human skills. The precedence restrictions between tasks can be expressed graphically using a precedence graph that contains a node for each task (each node has a corresponding weight representing the task execution time) and arcs to express precedence relationships: each arc $(u,v)$ indicates that task $v$ can not be started before finishing task $u$. A sample precedence graph with $n=11$ tasks is shown in Figura 1.

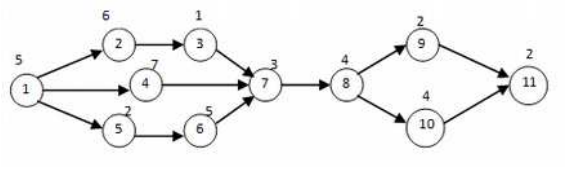


**Figure 1**. Example of precedence graph in modelling an instance of the ALB Problem

In [2] and [18] the following assumptions for ALB problems are defined:

- all operating parameters of the assembly line must be known at the time of its design;
- the assembly line is operated with a cycle time, i.e. the maximum processing time available for each work cycle;
- a task cannot be split among two or more workstations;
- the allocation of tasks to workstations should respect technological precedence requirements; though all tasks must be processed;
- tasks can be assigned to any workstation; all workstations have technological capacity to process any task;
- task processing times are independent of the workstation at which they are performed and of the preceding or following tasks;
- each station can process its assigned tasks within the given cycle time;
- any task can be processed at any workstation;
- the line is serial and processes an unique model of a single product.

Considering the characteristics of the line and according to the optimization objective considered, two major classes of ALB Problems can be identified in literature [2, 18]:

1) Simple Assembly Line Balancing Problem (SALB)

SALBP 1: the objective is to minimize the number of workstations for:

- A given a cycle time $C_T$ or
- A given working rhythm of the line, $R_T$, where $R_T=1/C_T$.

and the dual problem:

SALBP-2: Given the number of workstations m, minimize the cycle time $C_T$.

2) Generalized Assembly Line Balancing Problem (GALBP) with different formulations, which take into account further restrictions and other attributes of the simple problem.

Studies described in [1], [3] and [6] contain a comprehensive review of the literature related to ALB problems and classification schemes according to specific objectives.

# 3. Mathematical Formulation for the Multiple-Objective SALB-1 Problem

In practical applications, there is often a necessity to optimize a solution over multiple objectives. One of the advantages of using novel metaheuristic algorithms for the ALB problems is the ease of handling different objective functions. As a result, these approaches have been further explored by researchers, mainly to cope with the multiple objectives for these problems [13].

In this paper, a new method for balancing an assembly line is proposed: a fuzzy inertia-adaptive Particle Swarm Algorithm is used as the optimization tool to solve a multi-objective SALB-1 Problem. As in [7] and [22], three objectives are simultaneously considered:

- maximization of the line efficiency;
- minimization of the number of workstations actually used;
- minimization of the workload variation.

The problem can be described in mathematical programming as follows [10]:

## 3.1 Notations used to model the problem:

indices: $i$ for workstations and $j$ for tasks;

$m$: number of workstations actually used; $i=1,...,m$

$n$: number of tasks; $j=1,...,n$

$M$: number of available workstations; $M \leq n$

$t_j$: processing time of task $j$, i.e. the time required by task $j$ for completion;

$W$: total processing time;

$C_T$ (cycle time) – time interval between processing two consecutive production units;

$S_i$ : subset of all tasks assigned to workstation $i$ (workstation load);

$t(S_i)$ = workstation time: the sum of the times of all tasks assigned to workstation $i$;

$$t(S_i)=\sum_{j\in Si} t_j, \forall i \tag{1}$$

$PD(j)$ : the set of direct predecessors of task $j$;

$SC(j)$ : the set of direct successors of task $j$;

Notes:

1. For a given cycle time, $C_T$, a line balance is feasible only if the station time of neither station exceeds $C_T$.
2. The maximum execution time of each station is equal to the production rate $R$: $R= 1/C_T$ units of product per time unit.
3. If $t(S_i) < C_T$, then station $i$ has an idle time of $(C_T - t(S_i))$ time units in each cycle.

### 3.2 Decision variables:

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

### 3.3 Objective functions:

$$\max E=\frac{1}{m\cdot C_T}\sum_{j\in S_i} t_j\cdot x_{ij} \tag{2}$$

(maximization of line efficiency)

$$\min m=\sum_{i=1}^{M}\max_{1\leq j\leq n}\{x_{ij}\} \tag{3}$$

(minimization of the number of workstations actually used)

$$\min V=\frac{1}{m}\sum_{i=1}^{m}\left[t(S_i)-\frac{W}{m}\right]^2 \tag{4}$$

(minimization of workload variation)

### 3.4 Constraints:

$$\sum_{i=1}^{M} x_{ij}=1, \forall j \tag{5}$$

(every task $j$ is assigned to one and only one workstation)

$$\sum_{i=1}^{M} i\cdot x_{ik}\leq\sum_{i=1}^{M} i\cdot x_{ij}, \forall j, \forall k\in PD(j) \tag{6}$$

(the precedence constraints)

$$\sum_{j\in Si} t_j=t(S_i)=\sum_{j=1}^{n} t_j\cdot x_{ij}\leq C_T, \forall i \tag{7}$$

(the sum of the processing times of the tasks assigned to workstation $i$ does not exceed the cycle time)

## 4. Particle Swarm Optimization Algorithm principle

PSO is a metaheuristic optimization algorithm that is inspired by a social behaviour: the cooperative interaction among individuals within a swarm. Particles represent potential solutions to the optimization problem. They follow a global movement in their environment (the search space) while observing local movements in their neighbourhood. If the search space is $n$-dimensional, the state of the $i^{th}$ particle of the swarm is characterized by two $n$-dimensional vectors: position and velocity (speed). The quality of a particle's position is expressed by the particle's fitness value. This value is assessed according to the optimization function of the problem.

At each iteration of the algorithm, the state of each particle is updated using two different extreme values: $p_{best}$ and $g_{best}$. The first best value, $p_{best}$, is the best fitness function value that was recorded by that particle along its evolution (its individual experience). The second one, $g_{best}$, is the highest recorded value of the neighbourhood population (the collective experience).

In a $n$-dimensional search space, $S^n$, at time $t$, each particle $i$ has a position $x_i^t$ and it moves with a speed $v_i^t \in S^n$, according to its perception of the environment, based on the components presented above. At the beginning of the algorithm, the swarm is distributed randomly in the search space, each particle having a random position and speed. After initialization, the iterative optimization process is carried out: at iteration $t + 1$, positions and velocities of the particles are modified according to the formulas below [20]:

$$v_{ij}^{t+1}=\omega\cdot v_{ij}^t+c_1\cdot\varphi_1^t\cdot(y_{ij}^t-x_{ij}^t)+c_2\cdot\varphi_2^t\cdot(y¿tot_j^t-x_{ij}^t) \tag{8}$$

$$x_{ij}^{t+1} = x_{ij}^{t} + v_{ij}^{t+1} \quad (9)$$

where:

$j=1,...,n$ // $n$ = dimension of the decision variables

$i=1,...,PS$ // $PS$ = population size, i.e. the swarm dimension

- $x_{ij}^t$ is the position (decision variable) for dimension $j$ of particle $i$ at iteration $t$;
- $v_{ij}^t$ is the velocity for dimension $j$ of particle $i$ at iteration $t$;
- $y_{ij}^t$ is the personal best position $p_{best}$ of dimension $j$ attained by particle $i$ so far (the position giving the best fitness value);
- $y\_tot_j^t$ is the global best position $g_{best}$ of dimension $j$ reached by the particles of the swarm. This study refers to the global version of PSO, where the neighbourhood of a particle consists of all particles of the swarm.
- parameters $\varphi_1^t$ and $\varphi_2^t$ are random numbers uniformly distributed in the interval $[0,1]$, that are generated at every iteration;
- coefficients $c_1$ and $c_2$ are acceleration constants that contribute to self-learning (individual experience) and to the global movement (collective experience).
- parameter $\omega$ is the inertia factor; it is a scaling factor associated with the velocity in the previous time step. The weight of this factor defines the exploration-exploitation compromise: smaller values determine the decrease of particle velocity ⇒ more exploitation, while higher values determine the increase of particle velocity ⇒ more exploration.

Each particle's velocity at dimension $j$ is bounded by the user defined range $[-v_{max}, v_{max}]$.

A pseudocode version of the standard PSO algorithm is shown below [20].

During the initialization process, the following parameters are set by user:

1. The max iteration counter: $g_{max}$ (if this is the convergence criterion);
2. The total number of particles in the swarm: $PS$;
3. The values of the coefficients $c_1$, $c_2$ and $\omega$;
4. The maximum speed;

**for** $i=1,PS$
  initialize $x_i$ randomly; /* randomly generate the particle dimension between a minimum and a maximum value */
  initialize $v_i=0$; /*the initial velocity vector is zero for all particles*/
**while** (convergence criterion has not been met)
  **for** $i=1,PS$
    evaluate $f(x_i)$;//fitness value of a particle
    //update personal best:
    **if** $f(x_i^{t+1}) \geq f(y_i^t)$ then $y_i^{t+1} \leftarrow y_i^t$
      else $y_i^{t+1} \leftarrow x_i^{t+1}$;
    **endif**
    //update social optimum:
    $y\_tot^t \leftarrow min_{recorded\ in\ N(i)}\{f(y_i^t)\}$;
    update the velocity and position for each dimension $j$ of particle $i$ based on the updated values calculated from (8) and (9)
  **repeat**
**repeat**

# 5. Proposed Algorithm with Fuzzy Inertia Weight Controller

One disadvantage of the standard PSO algorithm is the lack of diversity and the probability of being trapped in local optima. In addition, the use of parameters with fixed values contradicts the collaborative search paradigm. This is an adaptive process so that different parameter values can be optimal only at certain stages of the search process. This means that the use of static parameters may lead to lower performance of the algorithm. One way of improving the algorithm was the concept of inertia. This concept was not included in the original formulation of the PSO algorithm [12]. The concept was developed in [20] and [21] to better control exploration and exploitation.

Because it affects the exploration-exploitation equilibrium, the inertia weight has attracted interest of researchers. Over time, different inertia weight strategies for particle swarm optimization have been developed to facilitate both global exploration and local exploitation during the optimization process.

The use of fuzzy controllers for tuning inertia weight was motivated by the necessity of solving two important issues that may experience a PSO algorithm: very small speed and premature convergence.

The proposed fuzzy controller for adaptive configuration of inertia weight is based on research results in literature, which showed that inertia weight adjustment in accordance with the current state of the optimization process can significantly improve the solution obtained and the convergence of the algorithm.

According to the main components of a fuzzy logic controller: 1.Fuzzification block, 2.Knowledge base, 3.Decision making block, 4.Defuzzification block, the proposed fuzzy controller is characterized as follows:

- Fuzzy sets defined with the triangular membership functions for each input and output variables;
- Fuzzification using continuous universe of discourse;
- Mamdani's „min" implication;
- De-fuzzification using the „centroid of area" technique.

Two input variables were selected as input to the fuzzy controller:

- current inertia weight $\omega^t$;
- a statistic expressing the current state of search: the coefficient of variance (normalized deviation of particles' fitness values):

$$DEV^t_{norm} = \frac{\sqrt{\frac{1}{PS}\sum_{i=1}^{PS}(p[i].val^t_f - \overline{val^t_f})^2}}{\overline{val^t_f}} \quad (10)$$

where:

$$\overline{val^t_f} = \frac{\sum_{i=1}^{PS} p[i].val^t_f}{PS} \quad (11)$$

is the average of fitness values recorded in the current swarm.

The range of values for the coefficient is *[0,1]*.

According to the study presented in [8] these input variables have five fuzzy linguistic degrees (*VL* – very low, *L* – low, *M* - medium, *H* – high, *VH* – very high) with associated membership functions of type left triangle, triangle and right triangle.

The membership functions for the inputs are described in Figure 2.

Table 1 presents the critical parameters $x_1$ and $x_2$ for the membership functions of the inputs.

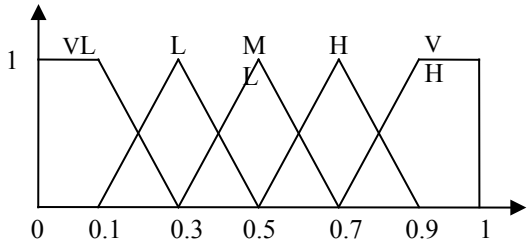


**Figure 2.** The membership functions for the inputs

**Table 1.** Values of the critical parameters for the inputs' membership functions

| | | $x_1$ and $x_2$ | |
|---|---|---|---|
| $\omega^t$ $DEV^t_{norm}$ | Left Triangle | 0 | 0.3 |
| | Triangle | 0.1 | 0.5 |
| | Triangle | 0.3 | 0.7 |
| | Triangle | 0.5 | 0.9 |
| | Right Triangle | 0.7 | 1 |

The output variable is the correction of the inertia weight, $\Delta\omega$.

The universe of discourse of the output variable is divided into three linguistic values (*D* – decrease, *UM* – unmodified, *I* - increase). Both positive and negative corrections are allowed for the inertia weight in the range of *[-0.1, 0.1]*.

The associated membership functions of the output are described in Figure 3 and Table 2 present the critical parameters $x_1$ and $x_2$ for the membership functions of the output.

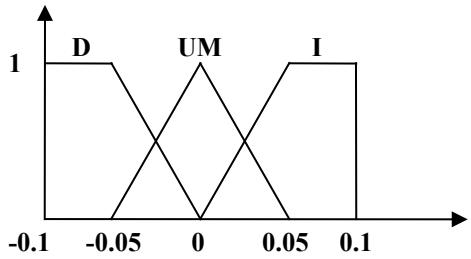


**Figure 3.** The membership functions for the output

**Table 2.** Values of the critical parameters for the output' membership functions

| | | $x_1$ and $x_2$ | |
|---|---|---|---|
| $\Delta\omega$ | Left Triangle | -0.1 | 0 |
| | Triangle | -0.05 | 0.05 |
| | Right Triangle | 0 | 0.1 |

Considering that in this problem the positions of the particles are directly related to their fitness values, the coefficient $DEV^t_{norm}$ expresses the distribution of particles within the swarm. A large value of this coefficient implies that the particles' locations are widely spread $\Longrightarrow$ a divergent swarm, whereas a small value of this coefficient implies that the particles' locations are close$\Longrightarrow$ a convergent swarm.

Once the correction of the inertia weight is calculated based on the fuzzy system, the inertia weight of the next iteration is adjusted as follows:

$$\omega^t = \omega^{t-1} + \Delta\omega \quad (15)$$

From the use of these statistics in the literature, the following conclusions can be drawn: when the coefficient of variance is high or very high, the individual particles are far away from each other. Then, if the current inertia weight is low or very low, its value should increase to achieve a global exploration.

If the current inertia weight has a medium value, its value should remain the same, whereas if the current inertia weight is high or very high, its value should decrease in order to balance the ability of particles to exploit and explore the search area.

Conclusions corresponding to the other situations are expressed in the corresponding fuzzy rules.

The inference table is presented below:

**Table 3.** The inference table

| $DEV^t_{norm}$ \ ω | VL | L | M | H | VH |
|---|---|---|---|---|---|
| VL | UM | UM | UM | D | D |
| L | UM | UM | D | D | D |
| M | I | UM | UM | UM | D |
| H | I | I | UM | UM | D |
| VH | I | I | UM | D | D |

# 6. Proposed PSO Algorithm for the Multiple-Objective SALB-1 Problem

The algorithm used in this paper for handling multiple objectives in the ALB problem is based on a multi-swarm approach of the PSO algorithm, namely Vector Evaluated Particle Swarm Optimization–VEPSO [16] when each objective function is optimized by a corresponding swarm; this swarm performs a PSO independently for its associated objective function using $y\_tot^t$ from another swarm. More precisely, the velocity update of the $m^{th}$ swarm (corresponding to the $m^{th}$ objective function) uses $y\_tot^{t(k)}$ from swarm $k$ as follows [14]:

$$v_{ij}^{t+1(m)} = \omega \cdot v_{ij}^{t(m)} + c_1 \cdot \varphi_1^t \cdot \left( y_{ij}^{t(m)} - x_{ij}^{t(m)} \right) + c_2 \cdot \varphi_2^t \cdot \left( y\_tot_j^{t(k)} - x_{ij}^{t(m)} \right) \quad (16)$$

$M$ = number of objective functions in the problem

$$k = \begin{cases} M, \text{ if } m = 1 \\ m\text{-}1, \text{ otherwise} \end{cases}$$

The initial population of particles (the initial swarm) is divided into three sub-populations of equal size (*3* = no. of objectives) according to a proportionate selection, that performs consecutively for each objective:

$$p[i].probability = \frac{f_{\max} - p[i].val\_f}{\sum_{r=1+(k-1)\cdot q}^{k\cdot q} (f_{\max} - p[r].val\_f)} \quad (17)$$

**let** $q=PS/3$;

**for** $k=1,..,3$ //*for every objective function*

**for** $i=1+(k-1)\cdot q,...,k\cdot q$

$p[i].val\_f = objective\_function_k$;

Calculate $f_{\max} = \max_{i=1+(k-1)\cdot q}^{k\cdot q} \{ p[i].val\_f \}$

Calculate $p[i].probability$;

**repeat**

based on these probabilities, $q$ solutions are selected by competitional selection; these solutions will form the $k^{th}$ sub-population;

**repeat**

To eliminate the disadvantage related to the fact that a non-dominated individual within a generation may become dominated in a later generation, the algorithm handles two archives:

- an archive in which the current generation non-dominant individuals are saved;
- an archive in which non-dominated individuals identified until the current time search are saved;

In the new generation, at iteration *(t+1)*, a percentage of the population will be replaced randomly with solutions from this external archive.

## 6.1 Solution representation

One of the most important issues in solving an ALB problem is to develop a good encoding scheme in order to obtain feasible balancing solutions, i.e. assignments of tasks to workstations so that the precedence constraints (the precedence relations specified in the corresponding precedence graph) are verified.

Since ALB problem is a discrete optimization problem, a discrete PSO algorithm must be used in order to deal with discrete variables. Different studies in literature implemented

discrete PSO algorithms for other production control optimization problems: [4], [9], [11]. A comprehensive survey on PSO algorithm and its applications is presented in [24].

A particle *p[i], i = 1, .., PS* defines a feasible sequence of tasks, being implemented as a structure comprising:

- *val_f,* the corresponding value of the objective function, obtained after the assignment of tasks to workstations;
- for every task $j \in \{1,...,n\}$ of the sequence:
  - the task velocity: *v[j]* and its personal best position: *y[j]*
  - the task priority; initially priorities are generated as random integers from *[1,n]*;
  - the task number, the task corresponding execution time, and the workstation to which is assigned.

Each component *v[j]* of the velocity vector will be initialized to *0*, then ranging between $[-v_{max}, v_{max}]$, with $v_{max}=n-1$.

### 6.2 Choice of PSO parameters

Swarm size *PS*: a common choice recommended in literature for small to medium size optimization problem is *PS = 20* to *60*. Accordingly, this study implements a swarm size *PS=60*.

Acceleration constants $c_1$ and $c_2$: this study follows a parameter automation strategy: time varying accelerator coefficients–TVAC (described in [17]) in order to enhance the global search at the beginning of the optimization process and to encourage the particles to converge quickly to the global optima in later stages:

$$c_1 = (0.5-2.5)\frac{gen}{gen_{max}} + 2.5$$
$$c_2 = (2.5-0.5)\frac{gen}{gen_{max}} + 0.5 \quad (18)$$

The convergence criterion: a given number of iterations; the maximum number of iterations, $gen_{max}$, was set to $gen_{max} = 50$, since higher values did not lead to an increase in algorithm performance.

## 7. Description of Test Problem and Computational Results

The typical PSO algorithm, the COMSOAL algorithm and the proposed algorithm were tested and compared in a computational study involving 10 problem instances of the standard ALB data sets from Scholl [19]: medium and large-scale benchmark problems with 11, 21, 29, 35 and 45 tasks, respectively and two different cycle time values. The results of the experiments showed better performance of the proposed algorithm in all cases.

Figure 4 presents the precedence graph of the largest multi-objective SALBP1 problem tested

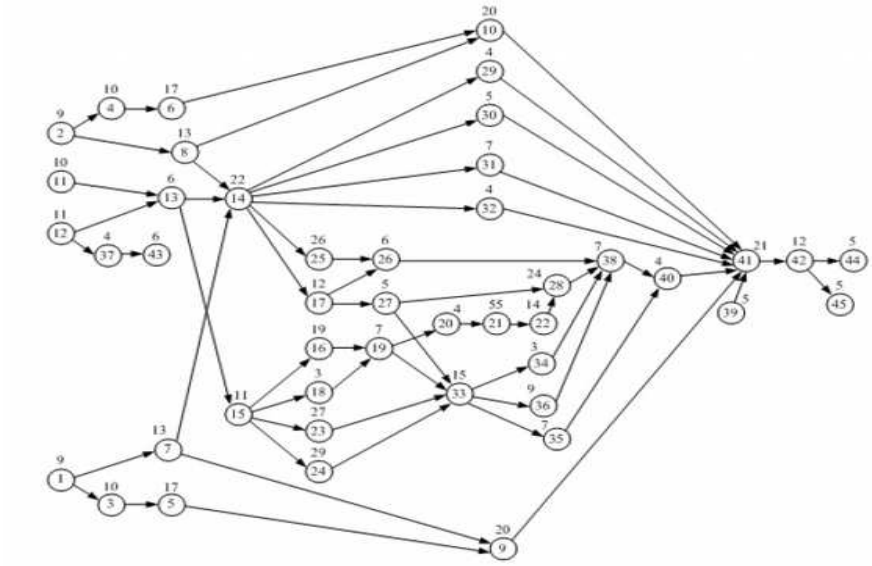

**Figure 4.** Precedence graph of the considered numerical example: „Kilbridge" problem with 45 tasks

(with 45 tasks) and Figure 5 describes the solution obtained for this problem considering a cycle time $C_T = 80$ minutes.

An additional criterion for assessing the results achieved by the proposed algorithm is the uniformity index. The uniformity index, *SI*, expresses the uniformity of balancing and is calculated as [10]:

$$SI = \sqrt{\frac{\sum_{i=1}^{m} [t(S)_{\max} - (t(S_i)]^2}{m}}, \quad (20)$$

where

$$t(S)_{\max} = \max_{1 \le i \le m} \{t(S_i)\} \quad (21)$$

Table 4 presents the results obtained by this method, compared to [22] and COMSOAL method (Computer Method of Sequencing Operations for Assembly Lines) of WinQSB software.

Notice: The algorithm generates another 3 different solutions with the same optimal values of objective functions, i.e.: No. of workstations actually used=7; Line efficiency = 98.5714%; Workload variation=0.693878 and the additional criteria: Total idle time=8; Uniformity index=1.41421.

C++ is used to realize the PSO algorithm. The interface is written in C# using Microsoft Visual Studio 2010. The average computation time of the algorithm is 15 seconds.

The configuration of computer is CPU Intel(R) Core(TM)2 Duo, CPU Clock Speed 2.20GHz, 4.00 GB RAM, and Windows 7 operating system.

As it can be observed in Table 4, the proposed algorithm is able to provide efficient solutions within reasonable computational time.

# 8. Conclusions

A multi-objective PSO approach combined with a fuzzy controller has been proposed to achieve optimal solutions for a multi-objective SALB-1 problem. The fuzzy module is applied to fine-tune dynamically the inertia weight in order to improve the performance of the PSO algorithm. The proposed algorithm minimizes the number of workstations for a given cycle time, minimizes the workload variation and furthermore maximizes the line efficiency.

The algorithm is easy to implement and is able to solve large test instances without additional computational complexity. It generates a set of optimal solutions and allows a decision to be made by the decision maker. The computational results and comparisons confirmed that the proposed algorithm is effective and efficient.

The hybridization of fuzzy and swarm intelligence technique supports a wide range of applications in industrial engineering and provides solutions to current Assembly Line problems. The extension of the proposed multi-objective SALB 1 model through the inclusion of other features which are typical for real-

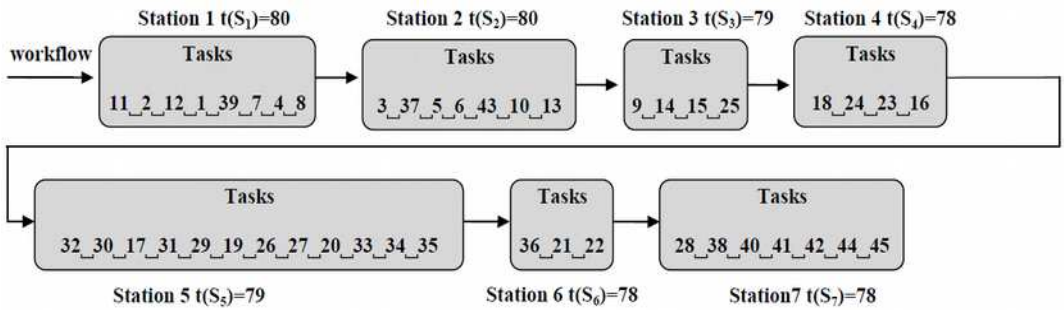


**Figure 5.** Solution obtained by the proposed approach for the Kilbridge problem

**Table 4.** Results obtained by the proposed algorithm

| | No. of workstations actually used | Line efficiency | Total idle time | Workload variation | Uniformity index |
|---|---|---|---|---|---|
| Proposed method | 7 | 98.57% | 8 min | 0.69 | 1.41 |
| [22] | 8 | 89.61% | 88 min | 84.25 | N.A. |
| COMSOAL | 8 | 89.61% | 88 min | 113 | N.A. |

world industrial applications (stochastic task times, workstation length restrictions, multiple equipments, etc.) necessitates further research.

Additions to the algorithm should include integration of other assembly process engineering issues, from the design phase of a product to the complex assembly building process.

# REFERENCES


1. AMEN, M., **Heuristic Methods for Cost–oriented Assembly Line Balancing: A Comparison on Solution Quality and Computing Time**, Int. J. of Production Economics, Vol. 69, 2001, pp. 255-264.
2. BAYBARS, I., **A survey of exact algorithms for SALBP,** Management Science, Vol. 32, 1986, pp. 11-17.
3. BETANCOURT, L. C., **ASALBP: the Alternative Subgraphs ALBP, Formalization and Resolution Procedures**, Doctoral thesis, Technical University of Catalonia, 2007.
4. BOUKEF, H., BENREJEB, M., BORNE, P.: **Flexible Jobshop Scheduling Problems Resolution Inspired From PSO**, Studies In Informatics And Control, Vol. 17(3), 2008, pp.241-252.
5. BOUTEVIN, C., GOURGAND, M. and NORRE, S., **Bin Packing Extensions for Solving an Industrial Line Balancing Problem,** Proceedings of the 5th IEEE International Symposium on Assembly and Task Planning, France, 10-11, 2003.
6. DELICE, Y., AYDOGAN, E.K., ÖZCAN, U., ILKAY, M.S., **A modified PSO algorithm to mixed-model two sided assembly line balancing**, Journal of Intelligent Manufacturing, 2014, pp. 1-22.
7. DELORME, X., BATTAÏA, O., DOLGUI, A., **Multi-objective Approaches for Design of Assembly Lines**, Multi-criteria and Game Theory Applications in Manufacturing and Logistics, Springer, 2014, pp. 31-56.
8. DINU, S. and POMAZAN, C.C., **A New Hybrid Fuzzy G.A. Optimization Method For Dynamic Economic Dispatch With Valve-Point Loading Effects,** MEQAPS '13, 2013, pp. 217-222.
9. DURAN, O. and PEREZ, L., **Solution of the spare parts joint replenishment problem with quantity discounts using a discrete particle swarm optimization technique**, Studies in Informatics and Control, Vol. 22(4), 2013, pp. 319–328.
10. GEN, M., CHENG, R. and LIN, L.: **Network Models and Optimization: Multiobjective Genetic Algorithm Approach**, Springer, Heidelberg, 2008.
11. GUILLERMO CABRERA, G., SILVANA RONCAGLIOLO, D., RIQUELME, J.P., CUBILLOS, C. and SOTO, R., **A hybrid PSO simulated annealing algorithm for the probabilistic travelling salesman problem**, Studies in Informatics and Control, vol. 21(1), 2012, pp. 49–58.
12. KENNEDY, J. and EBERHART, R.C., **Particle Swarm Optimization**, Proc. of the IEEE International Conference on Neural Networks, 1995, pp. 1942-1948.
13. LEVITIN, G., RUBINOVITZ, J. and SHNITS, B., **A genetic algorithm for robotic assembly line balancing**, European Journal of Operational Research, Vol. 168(3), 2006, pp. 811–825.
14. LIM, K.S., BUYAMIN, S., AHMAD, A., NAWAWI, S.W., IBRAHIM, Z., NAIM, F., GHAZALI, K.H. and MOKHTAR, N., **An improved VEPSO algorithm for multi-objective optimization problems**, Malaysia Japan Academic Scholar Conference (MJASC201), Springer, 2013.
15. MATONDANG, M. and JAMBAK, M.I., **Soft computing in optimizing assembly lines balancing,** Journal of Computer Science, Vol. 6, 2010, pp.141-162.
16. PARSOPOULOS, K.E. and VRAHATIS, M.N., **Particle swarm optimization method in multi-objective problems**, Proceedings of the 2002 ACM symposium on applied computing, 2002, pp.603–607.
17. RATNAWEERA, A., HALGAMUGE, S.K. and WATSON, H.C., **Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients**, IEEE Transactions on Evolutionary Computation, Vol. 8(3), 2004, pp. 240-255.
18. SCHOLL, A., **Balancing and Sequencing of Assembly Lines**, Physica, Heidelberg, 1999.

19. SCHOLL, A., **Data of assembly line balancing problems**, http://alb.mansci.de, Accessed: March 2015.

20. SHI, Y.H. and EBERHART, R.C., **Empirical study of particle swarm optimization**, Proceedings of the IEEE International Conference on Evolutionary Computation, Washington, DC, USA, 1999, pp. 1945–1950.

21. SHI, Y. H. and EBERHART, R.C., **A modified particle swarm optimizer**, Proceedings of the IEEE International Conferences on Evolutionary Computation, Anchorage, Alaska, USA, 1998, pp. 69–73.

22. SUWANNARONGSRI, S. and PUANGDOWNREONG, D., **Optimal Assembly Line Balancing Using Tabu Search with Partial Random Permutation Technique**, Int. Journal of Management Science and Engineering Management, Vol. 3(1), 2008, pp. 3-18.

23. YAZDANPARAST, V. and HAJIHOSSEINI, H., **Multi-manned production Lines with labour Concentration,** Australian Journal of Basic and Applied Sciences, Vol. 5(6), 2011, pp. 839-846.

24. ZHANG, Y., WANG, S., and JI, G., **A Comprehensive Survey on PSO Algorithm and Its Applications**, Mathematical Problems in Eng., Article ID 931256, in press, 2015.