Decision Science Letters 6 (2017) 1–10

Contents lists available at GrowingScience

Decision Science Letters

homepage: www.GrowingScience.com/dsl

A multi-objective method for solving assembly line balancing problem

Hadi Pazoki Toroudi[a*], Mahsa Sadat Madani[b], Fatemeh Sarlak[c] and Yosef Gholipour Kanani[d]

[a]*Young Researchers and Elite Club, Qaemshahr Branch ,Islamic Azad University ,Qaemshah, Iran*
[b]*Department of Industrial Engineering, TaJan University, Ghaemshahr, Iran*
[c]*Department of Industrial Engineering, College of Engineering, University of Tehran, Tehran, Iran*
[d]*Department of Industrial Engineering, Qaemshahr branch, Islamic Azad University, Qaemshahr, Iran*

C H R O N I C L E

Article history:
Received February 25, 2016
Received in revised format:
March 28, 2016
Accepted August 22, 2016
Available online
August 23 2016

Keywords:
Balancing assembly lines
Cycle time
Simulated Annealing
MOPSO

A B S T R A C T

Modeling the simple assembly line balancing (SALB) problem has covered a wide range of real-world applications. The recent advances in optimization problems have created the opportunities to tackle more challenging problems. This paper presents a multi-objective decision making problem to consider two objectives, cost and cycle time, for simple assembly line balancing. The problem is formulated as a mixed integer nonlinear optimization and the proposed study of this paper uses two metaheuristics to solve the resulted problem on some benchmark problems. The preliminary results have indicated that multi objective particle swarm optimization (MOPSO) has provided better quality solutions while the hybrid method based on MOPSO and simulated annealing has yielded more non-dominated Pareto solutions.

1. Introduction

An assembly line is a production system where the operational units executing the operations, called stations, are aligned in a serial operations (Gutjahr & Nemhauser, 1964). The workpieces pass different stations successively as they are gone forward along the line normally through some type of transportation system, e.g., a conveyor belt (Boysen et al., 2007). Assembly lines originally were developed for a mass-production of standardized products, designed to explore a high specialization of labor (Salveson, 1955; Gutjahr & Nemhauser, 1964). Ponnambalam et al. (2007) presented a multi-objective genetic algorithm like Chen et al. (2002) to solve assembly line balancing problems by considering the number of workstations, the line efficiency, the smoothness index before trade and transfer, and the smoothness index after trade and transfer and using genetic algorithm the problem was solved using different benchmarks (Scholl & Becker, 2006; Becker & Scholl, 2006)).

* Corresponding author.
E-mail address: h.pazoki.ums@gmail.com (H. Pazoki Toroudi)

doi: 10.5267/j.dsl.2016.8.005

Kottas and Lau (1973) proposed a heuristic for balancing paced assembly lines involving tasks which represent substantial time variations and by explicitly considering labor and incompletion costs in grouping tasks into work assignments to reach a near optimal solution. Kottas and Lau (1981) proposed a stochastic line balancing procedure and Sarin et al. (1999) extended this work by adding more assumptions. Gamberini et al. (2006) proposed a multi-objective heuristic algorithm for solving the stochastic assembly line re-balancing problem. Due to the high level of automation, assembly systems normally need significant amount of investment costs (Erel & Sarin, 1998). Therefore, the (re)-configuration of an assembly line pays essential role on cost efficient production system. Configuration planning normally includes all tasks and decisions which are associated with equipping the productive units for a given production process, before the actual assembly begins (Scholl & Scholl, 1999; Boysen et al., 2007).

There are literally various methods for solving SALB problem such as branch and bound (Pinto et al., 1975; Bukchin & Rabinowitch, 2006; Miralles et al. 2008; Ege et al., 2009) and dynamic programming (Henig, 1986; Bautista & Pereira, 2007). There are also several heuristics (Shin, 1990) and metaheuristics to solve this type problem including genetic algorithm (Rubinovitz & Levitin, 1995; Kim et al., 1996; Levitin et al., 2006; Tasan, & Tunali, 2008), Simulated annealing (Simaria & Vilarinho, 2001; Özcan, 2010; Seyed-Alagheband et al., 2011), tabu search (Lapierre et al., 2006), Ant Colony (Chica et al., 2011; McMullen & Tarasewich, 2003), differential evolution algorithm (Nourmohammadi & Zandieh, 2011; Zhang et al., 2016), There are also different techniques to tackle uncertainty associated with parameters in SALB (Reeve & Thomas, 1973; Andres et al., 2008; Silverman & Carter, 1986). Moreover, a class of SALB problem can be analyzed using simulation methods (Driscolla & Abdel-Shafi, 1985). SALB problem can also be investigated using different objectives (Kara et al., 2011; Chica et al., 2015; Sungur & Yavuz, 2015; Ramezanian & Ezzatpanah, 2015).

2. Problem definition

In a simple assembly line balancing (SALB), an assembly line is composed of $k = 1, ..., m$ (work) stations organized along a conveyor belt or other mechanical material handling devices. The jobs are moved through the line from station to station until they arrive the end of the line. A definite set of operations is executed over and over on any job, which enters a station, whereby the time span between two entries is called to as cycle time. Generally, the line balancing problem includes of optimally balancing the assembly work among all stations by optimizing different objectives. Thus, the total amount of work required to assemble a workpiece is split up into a set $V = \{1, ..., n\}$ of elementary operations named tasks. Tasks are unified units of work and thus each task j is related to a processing time t_j also referred to as task time. Because of technological and/or organizational requirements, tasks are not carried out in an arbitrary sequence, but are subject to precedence constraints. The usual input parameters of any SALB instance can be regularly represented by a precedence graph, which contains a node for each task, node weights which represents the task times and arcs take after direct as well as paths reflecting indirect precedence constraints. Fig. 1 demonstrates a simple precedence graph with $n = 9$ tasks containing task times between 2 and 9 (time units).

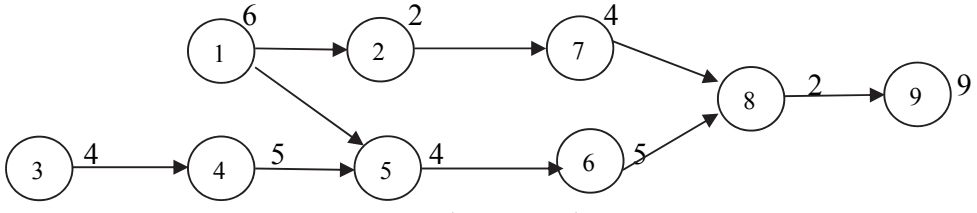

Fig. 1. Precedence graph

A possible *line balance*, i.e., an assignment of tasks to stations, needs to assure that all precedence relationships must hold. The set S_k of tasks assigned to a station k includes its station load or work

content, the cumulated task time $t(S_k) = \sum_{j \in S_k} t_j$, station time. SALB also considers that the cycle time of all stations is equal to the an arbitrary value, c. Moreover, all station times of a possible balance do not exceed c, as otherwise the required operations could not be completed before the workpiece exits the station. Station times maybe smaller than the cycle time, in which case a station *k* maintains an unproductive *idle time* of $c - t(S_k)$ time units in each cycle. In Fig. 1, a feasible line balance with cycle time c = 11 and m = 5 stations is represented by station loads $S_1 = \{1,3\}, S_2 = \{2,4\}, S_3 = \{5,6\}$, $S_4 = \{7,8\}, S_5 = \{9\}$.

2.1. Notations

The following notations are used for the proposed model of this paper.

i, i′	Index of operations
j	Index of work stations
s	Position of operation in work stations
n	Number of operations
m	Number of work stations
MO	Maximum number of operations, which could be assigned to a work station
Lt_i	Minimum time of operation *i*
Ut_i	Maximum time of operation *i*
P	Set the precedence relationships between operations such that for every $(i, i') \in P$ operations *i* is located right before *i′*
IP_i	Ser of all operations needed for accomplishment of task *i*
$ST_{i,i'}$	Setup time of operations *i′,* which is accomplished after operation *i*
EQC	The cost to equip each operation per workstation per unit of time

2.2. Variables

Index	Decscription	Variable type
t_i	Processing time of operation *i*	Real
CT	Cycle time	Real
NT_j	Number of operations assigned to work station *j*	Integer
TS_j	Processing time of work station *j*	Real
X_{ijs}	One if operation *i* in position *s* is processed in workstation *j,* and zero, otherwise	Binary
$Y_{ii'j}$	One if operation *i* is processed before operation *i′* in workstation j, and zero, otherwise	Binary

2.3. Mathematical model

Based on the assumption and notation given in this paper, we present the mathematical model of the paper as follows,

$$\min\ Z_1 = CT \tag{1}$$

$$\min\ Z_2 = \sum_{j=1}^{m} NT_j .EQC . \sum_{i=1}^{n} \sum_{s=1}^{MO} t_i^{-1} .x_{ijs} \tag{2}$$

subject to

$$\sum_{j=1}^{m} \sum_{s=1}^{MO} x_{ijs} = 1 \qquad \forall i \tag{3}$$

$$\sum_{i=1}^{n}\sum_{s=1}^{MO} x_{ijs} \geq 1 \qquad \forall j \tag{4}$$

$$\sum_{i=1}^{n} x_{ijs} \leq 1 \qquad \forall j, s \tag{5}$$

$$\sum_{i=1}^{n} x_{ij,s+1} - \sum_{i=1}^{n} x_{ij,s} \leq 0 \qquad \forall j, s = 1,...,MO-1 \tag{6}$$

$$\sum_{j=1}^{m}\sum_{s=1}^{MO} (MO(j-1)+s).x_{ijs} - \sum_{j=1}^{m}\sum_{s=1}^{MO} (MO(j-1)+s).x_{i'js} \leq 0 \qquad \forall (i, i') \in P \tag{7}$$

$$Lt_i \leq t_i \leq Ut_i \qquad \forall i \tag{8}$$

$$ST_j = \sum_{i=1}^{n}\sum_{s=1}^{MO} t_i.x_{ijs} + \sum_{i=1}^{n}\sum_{i'=1}^{n} ST_{ii'}.y_{ii's} \leq CT \qquad \forall j \tag{9}$$

$$x_{ijs} + x_{i'j,s+1} \leq 1 + y_{ii'j} \qquad \forall j, s \quad \forall (i, i') | (i \neq i') \wedge (i' \notin IP_i) \tag{10}$$

$$NT_j = \sum_{i=1}^{n}\sum_{s=1}^{MO} x_{ijs} \qquad \forall j \tag{11}$$

$$x_{ijs} = \{0,1\} \qquad \forall i, j, s \tag{12}$$

$$y_{ii'j} = \{0,1\} \qquad \forall i, i', j \tag{13}$$

$$CT \geq 0 \tag{14}$$

$$NT_j \in N \qquad \forall j \tag{15}$$

The first objective function given in Eq. (1) minimizes total cycle time and the second objective function given in Eq. (2) minimizes the total cost of equipping operations to different workstations. According to Eq. (3), each operation has to be assigned to one workstation. Eq. (4) assures that in each workstation, only one operation is accomplished. According to Eq. (5), for each position, one task is accomplished in one workstation. Eq. (6) is associated with the precedence of operations. Eq. (7) guarantees that in case in graph, task, precedence of *i'* are before operation *i,* then operations of *i* must be accomplished before task *i* and in case both tasks *i* and *i'* are accomplished in one workstation, then task *i* has to be executed in one of previous workstations. Eq. (8) is associated with the limitations on processing time. Eqs. (9-11) are associated with the processing times and Eqs. (12-15) determine the type of variables. The resulted problem formulation is a nonlinear mixed integer programming problem. SALB problem is an NP-Hard in its simplest form (Gutjahr & Nemhauser, 1964). Therefore, it is not possible to solve the problem for real-word problems in reasonable amount of time and we use two metaheuristics; namely multi objective particle swarm optimization (MOPSO) and simulated annealing (SA) to find some near optimal solutions.

3. MOPSO and SA Algorithms

Before describing MOPSO algorithm, the particle swarm optimization (PSO) needs to be presented, shortly. In PSO, first, some predefined particles are generated incidentally in the solution space. The situation of each particle represents an order of alternatives. Besides, the fitness function value for algorithm is given identically as the determined score for the considered problem. In each iteration, each particle should be moved based on other particles situations. Two types of solutions have to be updated in each iteration of this algorithm. The first variable which is denoted by *PBest* is the best permutation experienced by each particle while the second one named as *GBest* is the best experienced arrangement by all particles. Particles movement direction and their final position in each iteration will be calculated according to the following equations:

$$V_i(t+1) = w.V_i(t) + c_1 r_1\big(PBest_i(t) - X_i(t)\big) + c_2 r_2\big(GBest(t) - X_i(t)\big), \tag{16}$$

$$X_i(t+1) = X_i(t) + V_i(t+1). \tag{17}$$

where, $\boldsymbol{V_i(t)}$ is the velocity vector of i^{th} particle in the t^{th} iteration moment, $X_i(t)$ represents the i^{th} particles situation in t^{th} iteration, $\boldsymbol{w}$، $\boldsymbol{c_1}$ and $\boldsymbol{c_2}$ are the constant numbers which show the share of the particle's present situation, the best experienced position of a particle and the best experienced permutation of the whole particles, respectively. Finally, $\boldsymbol{r_1}$ and $\boldsymbol{r_2}$ are the incidental numbers between zero and one. The implementation of MOPSO is similar to the work by Bashiri and AliAskari (2014), Coello et al. (2004), Eberhart and Kennedy (1995), Kennedy (1998), Intelligence (2007) and Low et al. (2010) and interested readers could read that work. The proposed method of this paper combines the simulated annealing method given by Osman and Potts (1989) with MOPSO for the proposed method. The implementation of SA (Kirkpatrick, 1984; Osman & Potts, 1989) is as follows,

Hybrid MOPSO and SA

Initialize T, MAXITER, β , ITER=1
Do while Iter < MAXITER
i=0
Do until i=N or reach to Equilibrium
Generate neighborhood of current solution
Δ =f (neighborhood)-f (current solution)
If Δ >0
Current solution=neighborhood
Else
p=rand (0, 1)
If p < $\exp(-\Delta E / T)$
Current solution= neighborhood
End if
End if
i=i+1
Loop until
Iter=Iter+1
T=T/(1+ βT)
Loop while

The proposed hybrid MOPSO-SA used in this paper has the following steps,

1. Position and velocity of the particles are randomly generated in the solution space,
2. Compute the fitness of each member,
3. Update the position of each particle and the particles,
4. Update each particle's position and velocity,
5. Use SA to improve the solution,
6. Go to step 2 until it meets termination criteria.

3. The results

The proposed study of this paper uses some benchmark problems to measure the performance of the proposed study gathered from http://alb.mansci.de/files/uploads/SALBP data sets.zip library. The setup time is considered as follows,

$$ind = \frac{\sum_{i=1}^{n} (Ut_i + Lt_i)}{4.n}$$

In addition, the existing times are considered as lower time and upper bound is considered by multiplying this number by 5. To measure the performance of the proposed study, we use spacing matrix as follows (Deb, 2001),

$$S = \sqrt{\frac{1}{m}\sum_{j=1}^{m}(d_j - \bar{d})^2} \tag{18}$$

where $d_j = \min_{k \in n \& k \neq j} \sum_{i=1}^{n} |f_i^j - f_j^k|$ and $\bar{d} = \sum_{j=1}^{m} \frac{d_j}{|n|}$. We also use Mean Ideal Distance (MID) as follows,

$$MID = \frac{\sum_{j=1}^{m} C_j}{m} \tag{19}$$

All computations have been executed on a Laptop computer and the algorithms have been coded on MATLAB. Table 1 demonstrates the results of the proposed method. The third column of the table represents the number of operations. Columns four to six show the performance of the proposed MOPSO-SA and the results are compared with the performance of MOPSO presented in columns 7-9. For each method we present the number of Pareto solution (Nop), MID and Spacing. Fig. 1 shows two objective functions. Moreover Fig. 2, Fig. 3 and Fig. 4 compare the Spacing, MID and Nop measures.

Table 1
The results of the implementation of MOPSO versus MOPSO-SA

Prob.	Problem	Number of operations	MOPSO-SA			MOPSO		
			MID(10^7)	Nop	Spacing	MID(10^7)	Nop	Spacing
1	BOWMAN8	8	0.0043659	5	392	0.0045693	10	427
2	JAESCHKE	9	0.00294	7	139	0.003519	16	27.5
3	JACKSON	11	0.004147	7	262.6	0.0043406	14	63.3
4	HESKIA	28	0.1753	5	904.2	0.1683	12	2389
5	BUXEY	29	0.0334	8	1602	0.0356	25	3585
6	GUNTHER	35	0.0617	5	1097	0.0695	17	1307
7	KILBRID	45	0.087	6	848.8	0.122	26	4408
8	HAHN	53	0.107	7	611.9	0.212	15	1658
9	ARC83	83	0.1368	3	9063	0.24274	10	860.7
10	ARC111	111	0.227949	4	3478	0.38	15	3615
11	BARTHOL2	148	0.28682	4	2640	0.386215	20	4097
12	BARTHOLD	148	0.13136	5	1277	0.167024	12	1016

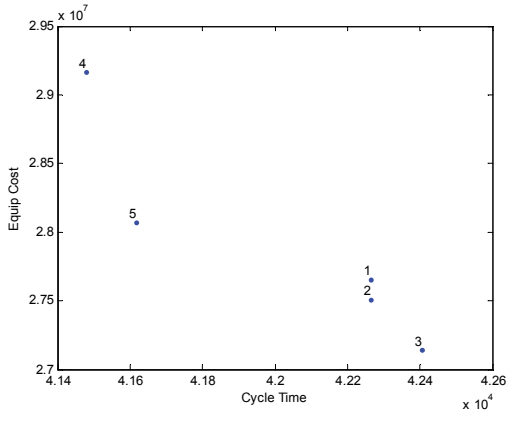

Fig. 1. Pareto solutions of Cycle time versus Equip cost

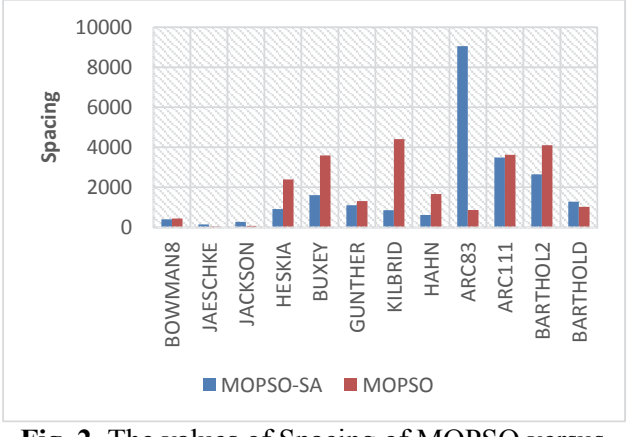

Fig. 2. The values of Spacing of MOPSO versus MOPSO-SA

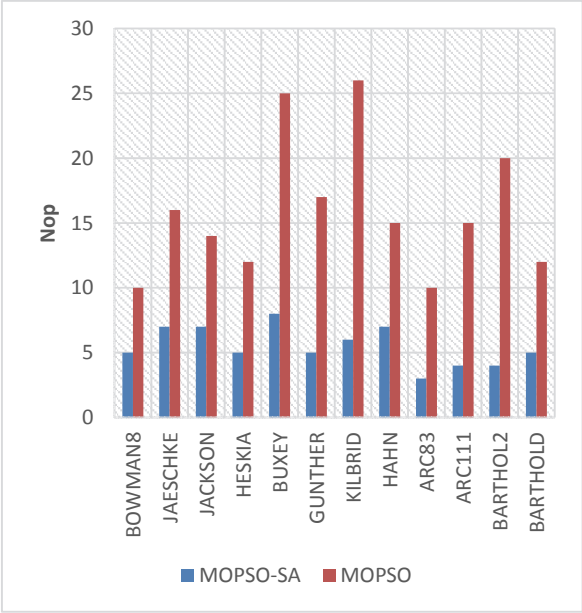

Fig. 3. The values of Nop of MOPSO versus MOPSO-SA

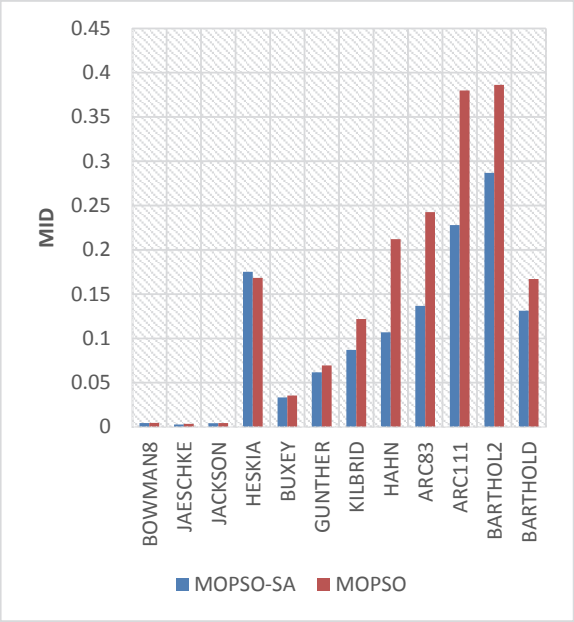

Fig. 4. The values of MID of MOPSO versus MOPSO-SA

As we can observe from the results of Figs. (2-4), the proposed MOPSO-SA has maintained better performance compared with MOPSO. Moreover, the results of Fig. 3, the hybrid method, MOPSE-SA has found fewer numbers of Pareto solution compared with MOPSO.

4. Conclusion

Modeling the SALB problem has covered a wide range of real-world applications. The recent advances of optimization problems have created the opportunities to solve more challenging problems. In this paper, we have presented a multi-objective decision making problem to consider two objectives for simple assembly line balancing; namely cycle time and cost. The problem has been formulated as mixed integer nonlinear optimization and the proposed study of this paper has implemented two metaheuristics to solve the resulted problem on some benchmark problems. The preliminary results have indicated that multi objective particle swarm optimization has provided better quality solutions while the hybrid method yielded more non-dominated Pareto solutions. The results of this paper are somewhat similar with the PSO method proposed by Nearchou (2011) and SA method proposed by Cakir et al. (2011).

Acknowledgement

The authors would like to thank the anonymous referees for constructive comments on earlier version of this paper.

References

Andres, C., Miralles, C., & Pastor, R. (2008). Balancing and scheduling tasks in assembly lines with sequence-dependent setup times. *European Journal of Operational Research, 187*(3), 1212-1223.

Bashiri, M., & AliAskari, E. (2014). A permutation decision making method with multiple weighting vectors of criteria using NSGA-II and MOPSO. *Decision Science Letters, 3*(2), 197-208.

Bautista, J., & Pereira, J. (2007). Ant algorithms for a time and space constrained assembly line balancing problem. *European Journal of Operational Research, 177*(3), 2016-2032.

Becker, C., & Scholl, A. (2006). A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research, 168*(3), 694-715.

Boysen, N., Fliedner, M., & Scholl, A. (2007). A classification of assembly line balancing problems. *European Journal of Operational Research, 183*(2), 674-693.

Bukchin Y, Rabinowitch I (2006) A branch-and-bound based solution approach for the mixed-model assembly line-balancing problem for minimizing stations and task duplication costs. *European Journal of Operational Research, 174*(1):492–508.

Cakir, B., Altiparmak, F., & Dengiz, B. (2011). Multi-objective optimization of a stochastic assembly line balancing: A hybrid simulated annealing algorithm. *Computers & Industrial Engineering, 60*(3), 376-384.

Chen, R. S., Lu, K. Y., & Yu, S. C. (2002). A hybrid genetic algorithm approach on multi-objective of assembly planning problem. *Engineering Applications of Artificial Intelligence, 15*(5), 447-457.

Chica, M., Cordón, Ó., Damas, S., & Bautista, J. (2015). Interactive preferences in multiobjective ant colony optimisation for assembly line balancing. *Soft Computing, 19*(10), 2891-2903.

Chica, M., Cordón, O., Damas, S., & Bautista, J. (2011). A new diversity induction mechanism for a multi-objective ant colony algorithm to solve a real-world time and space assembly line balancing problem. *Memetic Computing, 3*(1), 15-24.

Coello, C. A. C., Pulido, G. T., & Lechuga, M. S. (2004). Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation, 8*(3), 256-279.

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms* (Vol. 16). John Wiley & Sons.

Driscolla, J., & Abdel-Shafi, A. A. (1985). A simulation approach to evaluating assembly line balancing solutions. *International Journal of Production Research, 23*(5), 975–985.

Eberhart, R. C., & Kennedy, J. (1995, October). *A new optimizer using particle swarm theory*. In Proceedings of the sixth international symposium on micro machine and human science (Vol. 1, pp. 39-43).

Ege, Y., Azizoglu, M., & Ozdemirel, N. E. (2009). Assembly line balancing with station paralleling. *Computers & Industrial Engineering, 57*(4), 1218-1225.

Erel, E., Sarin, S.C., 1998. A survey of the assembly line balancing procedures. *Production Planning & Control 9*(5), 414–434.

Gamberini, R., Grassi, A., & Rimini, B. (2006). A new multi-objective heuristic algorithm for solving the stochastic assembly line re-balancing problem. *International Journal of Production Economics, 102*(2), 226-243.

Gutjahr, A.L., & Nemhauser, G.L. (1964). An algorithm for the line balancing problem. *Management Science 11*(2), 308–315.

Henig, M. I. (1986). Extensions of the dynamic programming method in the deterministic and stochastic assembly-line balancing problems. *Computers & Operations Research, 13*(4), 443-449.

Intelligence, S. (2007). Particle swarm optimization. MCCAFFREY, James.[online].[cit. 2014-05-20]. Dostupné z: http://msdn. microsoft. com/en-us/magazine/hh335067. aspx.

Kara, Y., Özgüven, C., Seçme, N. Y., & Chang, C. T. (2011). Multi-objective approaches to balance mixed-model assembly lines for model mixes having precedence conflicts and duplicable common tasks. *The International Journal of Advanced Manufacturing Technology, 52*(5-8), 725-737.

Kennedy, J. (1998, March). *The behavior of particles. In International Conference on Evolutionary Programming* (pp. 579-589). Springer Berlin Heidelberg.

Kim, Y. K., Kim, Y. J., & Kim, Y. (1996). Genetic algorithms for assembly line balancing with various objectives. *Computers & Industrial Engineering, 30*(3), 397-409.

Kirkpatrick, S. (1984). Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics, 34*(5-6), 975-986.

Kottas, J. F., & Lau, H. S. (1973). A Cost-Oriented Approach to Stochastic Line Balancing1. *AIIE Transactions, 5*(2), 164-171.

Kottas, J. F., & Lau, H. S. (1981). A stochastic line balancing procedure. *International Journal of Production Research, 9*(2), 177–193.

Lapierre, S. D., Ruiz, A., & Soriano, P. (2006). Balancing assembly lines with tabu search. *European Journal of Operational Research, 168*(3), 826-837.

Levitin, G., Rubinovitz, J., & Shnits, B. (2006). A genetic algorithm for robotic assembly line balancing. *European Journal of Operational Research, 168*(3), 811-825.

Low, C., Hsu, C. J., & Su, C. T. (2010). A modified particle swarm optimization algorithm for a single-machine scheduling problem with periodic maintenance. *Expert Systems with Applications, 37*(9), 6429-6434.

McMullen, P. R., & Tarasewich, P. (2003). Using ant techniques to solve the assembly line balancing problem. *IIE Transactions, 35*(7), 605-617.

Miralles, C., García-Sabater, J. P., Andrés, C., & Cardós, M. (2008). Branch and bound procedures for solving the assembly line worker assignment and balancing problem: Application to sheltered work centres for disabled. *Discrete Applied Mathematics, 156*(3), 352-367.

Nearchou, A. C. (2011). Maximizing production rate and workload smoothing in assembly lines using particle swarm optimization. *International Journal of Production Economics, 129*(2), 242-250.

Nourmohammadi, A., & Zandieh, M. (2011). Assembly line balancing by a new multi-objective differential evolution algorithm based on TOPSIS. *International Journal of Production Research, 49*(10), 2833-2855.

Osman, I. H., & Potts, C. N. (1989). Simulated annealing for permutation flow-shop scheduling. *Omega, 17*(6), 551-557.

Özcan, U. (2010). Balancing stochastic two-sided assembly lines: A chance-constrained, piecewise-linear, mixed integer program and a simulated annealing algorithm. *European Journal of Operational Research, 205*(1), 81-97.

Pinto, P., Dannenbring, D. G., & Khumawala, B. M. (1975). A branch and bound algorithm for assembly line balancing with paralleling. *International Journal of Production Research, 13*(2), 183-196.

Ponnambalam, S. G., Aravindan, P., & Naidu, G. M. (2000). A multi-objective genetic algorithm for solving assembly line balancing problem. *International Journal of Advanced Manufacturing Technology, 16*(5), 341-352.

Ramezanian, R., & Ezzatpanah, A. (2015). Modeling and solving multi-objective mixed-model assembly line balancing and worker assignment problem. *Computers & Industrial Engineering, 87*, 74-80.

Reeve, N. R., & Thomas, W. H. (1973). Balancing stochastic assembly lines. *AIIE Transactions, 5*(3), 223–229.

Rubinovitz, J., & Levitin, G. (1995). Genetic algorithm for assembly line balancing. *International Journal of Production Economics, 41*(1), 343-354.

Salveson, M. E. (1955). The assembly line balancing problem. *Journal of Industrial Engineering, 6*(3), 18-25.

Sarin, S. C., Erel, E., & Dar-El, E. M. (1999). A methodology for solving single-model, stochastic assembly line balancing problem. *Omega, 27*(5), 525-535.

Scholl, A., & Becker, C. (2006). State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *European Journal of Operational Research, 168*(3), 666-693.

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

Seyed-Alagheband, S.A., Fatemi Ghomi, S.M.T., Zandieh, M., 2011. A simulated annealing algorithm for balancing the assembly line type II problem with sequence-dependent setup times between tasks. *International Journal of Production Research 49*(3), 805–825.

Shin, D. (1990). An efficient heuristic for solving stochastic assembly line balancing problems. *Computers and Industrial Engineering, 18*(3), 285–295.

Silverman, F. N., & Carter, J. C. (1986). A cost-based methodology for stochastic line balancing with intermittent line stoppages. *Management Science, 32*(4), 455-463.

Simaria, A. S., & Vilarinho, P. M. (2001). The simple assembly line balancing problem with parallel workstations-a simulated annealing approach. *International Journal of Industrual Engineering-Theory and Applications, 8*(3), 230-240.

Sungur, B., & Yavuz, Y. (2015). Assembly line balancing with hierarchical worker assignment. *Journal of Manufacturing Systems, 37*, 290-298.

Tasan, S. O., & Tunali, S. (2008). A review of the current applications of genetic algorithms in assembly line balancing. *Journal of intelligent manufacturing, 19*(1), 49-69.

Zhang, H., Yan, Q., Liu, Y., & Jiang, Z. (2016). An integer-coded differential evolution algorithm for simple assembly line balancing problem of type 2. *Assembly Automation, 36*(3).