Proceedings of the 2012 International Conference on Industrial Engineering and Operations Management
Istanbul, Turkey, July 3 – 6, 2012

Meta-heuristic Approaches for the Assembly Line Balancing Problem

Adil Baykasoğlu
Department of Industrial Engineering
Dokuz Eylul University, Tınaztepe Campus, Izmir, Turkey

Şebnem Demirkol Akyol
Department of Industrial Engineering
Dokuz Eylul University, Tınaztepe Campus, Izmir, Turkey

Abstract

Assembly line balancing problem (ALBP) is a decision problem arising when designing or redesigning an assembly line and it consists in finding the optimal assignment of tasks among the workstations corresponding to some objectives. This paper deals with the single-model ALBP with the aim of minimizing the cycle time and maximizing the workload smoothness among the workstations. In order to solve the problem we propose two different meta-heuristic approaches, namely greedy randomized adaptive search procedure (GRASP) and particle swarm optimization (PSO). An industrial case study is included in order to compare the performance of the proposed techniques.

Keywords
Single-model assembly line balancing problem, particle swarm optimization, greedy randomized adaptive search procedure, meta-heuristics, multi-objective optimization

1. Introduction
In today's highly competitive global business environment there are many objectives for companies to cope with. This is one of the main reasons why multi-objective optimization problems usually occur. Assembly line balancing problems (ALBP) are one of the typical multi-objective optimization problems in industry. Since the first article on the ALBP (Salveson 1955), it has been a hot topic for researchers. ALBP is relevant for the allocation of the tasks, each having an operation processing time and a set of precedence relations, among workstations such that the precedence relations are satisfied and some measure of performance is optimized corresponding to some objectives such as maximizing the line efficiency, distributing the tasks evenly as possible to the workstations, minimizing the assembly costs, minimizing the probability of the tasks' lateness, etc…

According to the classification proposed by Scholl (1999) there are three fundamental types of ALBPs according to the product mix. If only one particular homogenous product is assembled at the assembly line, the problem is a single-model assembly line balancing problem (SALBP). In single-model assembly lines, one homogenous product or several products with identical production process are assembled. If more than one product is produced in batches, the problem becomes a multi-model ALBP. Finally, the last type is the mixed-model assembly line balancing problem which refers to assembly lines which are capable of producing a variety of similar product models simultaneously and continuously (not in batches).

In the relevant literature, assembly line balancing problems are also classified into two different types in accordance with the objective function of the optimization problem, ALBP-1 and ALBP-2. Type 1 problems try to minimize the number of workstations hence, the cycle time must be predetermined, and in industrial life this type of balancing problems is occurred when the designing phase of a new assembly line. On the other hand, type 2 problems try to minimize the cycle time for a fixed number of workstations, so this type of balancing problems is more appropriate

when redesigning an existing line. Each type of the SALBPs belongs to the NP-hard class of combinatorial optimization problems (Karp 1972; Scholl 1999).
In this paper we deal with a real-life ALBP that occurs at an electric company which is located in Manisa, Turkey. The company operates many lines, but one of them is much more critical because of the product's value. On that particular line a single-model industrial electric unit is assembled, and demand for the unit is increasing. Due to the layout and location constraints, company cannot duplicate the line or cannot open a parallel workstation. Under these circumstances, management wants to increase the production rate. Since, increasing the production rate means decreasing the cycle time, the assembly line should be rebalanced. This is the motivation of this paper. In this industrial case, the problem is a type 2 single assembly line balancing problem (SALBP-2).

A comprehensive review of SALB and its solution procedures was provided by Scholl and Becker (2006). It is stated that although there have been a great deal of interest on ALBPs, only few of them were focused on type 2 problems. Hackman et al. (1988) described various heuristic approaches to deal with both SALBP-1 and SALBP-2. Scholl and Voss (1996) proposed bidirectional and dynamic extensions to heuristic priority rules for both SALBP type 1 and 2. Besides, they developed improvement procedures for SALBP-2 and combined with tabu search. Klein and Scholl (1999) described a branch and bound procedure that directly solves SALBP-2 by using a new enumeration technique the Local Lower Bound Method, which is complemented by a number of bounding and dominance rules. Nearchou (2006) proposed a new population based heuristic to deal with the SALP-2. The author considered the following objectives: minimizing the cycle time, balancing delay time of the stations and maximizing the workload smoothness index of the line. He also developed a particle swarm optimization (PSO) algorithm for the proposed multi-objective problem in 2011 (Nearchou 2006). Seyed-Alaghebanda et al. (2011) addressed the general assembly line balancing problem where the simple version is enriched by considering sequence-dependent setup times between tasks. They applied a simulated annealing algorithm to deal with the proposed problem.

After the literature review it is observed that while much research is concerned with type 1 problems, type 2 problems have not been attracted much attention. In this research paper, a bi-objective single assembly line balancing problem of type 2 is considered. Two different meta-heuristics, greedy randomized adaptive search procedure (GRASP) and particle swarm optimization (PSO) are applied to solve a real-life assembly line balancing problem.

The remainder of the paper is organized as follows. Mathematical model of the addressed problem (SALBP-2) is described in section 2. The proposed GRASP and PSO approaches are defined in sections 3 and 4, respectively. The industrial case study is presented in section 5. Finally, conclusions are pointed out in section 6.

2. Mathematical Modeling of the Problem

The problem addressed in this research is the single-model assembly line balancing problem of type 2 (SALBP-2). The optimal assignment of tasks among the workstations should have been made in order to minimize cycle time for a predetermined number of workstations. The notation given by Scholl (1999) is used to express the mathematical model for the SALBP-2 in this study. The mathematical model of the problem has the following features:

- The line is operated with a cycle time c denoting the maximum processing time available for each work cycle.
- Each station can complete its assigned tasks within the specified cycle time.
- There are n tasks to be performed and V is the set of tasks $V = \{1, \ldots, n\}$.
- m is the predetermined number of workstations and WS is the set of workstations $WS = \{1, \ldots, m\}$.
- Each task i ($i \in V$) is processed on exactly one workstation with the operation time of t_i, and t_{sum} is the sum of all task times.
- S_z ($z \in WS$) is the idle time of workstation k corresponding to the difference between the cycle time c and - the summation of the processing times of the tasks being executed on workstation z.
- St_z ($z \in WS$) is the station time of workstation k indicating that the sum of task times of works being operated at station z.
- Each task cannot be operated before its all direct predecessors are completed, and this relationship is given by the precedence graph (precedence relations).

The decision variables are defined as follows (Simaria and Vilarinho 2004):

s_z = idle time of workstation z

Mathematical programming model for the SALBP-2 is given in the following (Nearchou 2011):

$$\text{Min} \quad c + \frac{m}{m-1}\sum_{z=1}^{m}\left(\frac{S_z}{\sum_{l=1}^{m} S_l} - \frac{1}{m}\right)^2 \tag{1}$$

subject to:

$$\sum_{z=1}^{m} x_{ik} = 1 \qquad i = 1, \ldots, n \tag{2}$$

$$\sum_{z=1}^{m} x_{ak} - \sum_{z=1}^{m} x_{bk} \leq 0 \qquad a \in V,\ b \in F_a \tag{3}$$

$$St_z \leq t_{sum} \qquad \text{for all } k = 1,\ldots,m \tag{4}$$

$$St_z \leq c \qquad \text{for all k} = 1,\ldots,m \tag{5}$$

$$c \geq t_{max} \tag{6}$$

where t_{max} is the maximum operation time of all tasks $t_{max} = \max\{t_1, t_2, \ldots, t_n\}$.

The objective function (1) is the minimization of the sum of two terms. The first term of the objective function minimizes cycle time (c) for predetermined number of workstations (m). Also, by minimizing the second term of the objective function, the workload is distributed among workstations as identical as possible. The primary objective of this bi-objective problem is to minimize cycle time, and the secondary one is to smoothen the workload. Because the value of the second term is within the range [0,1], the first term, namely the cycle time, is become dominant for c values greater than 1. In industrial life cycle time c is always greater than 1, so the model minimizes cycle time before the secondary goal.

The first set of constraints (2) ensures that each task is assigned to only one workstation. Inequality (3) states that no successor of a task is assigned to an earlier station than that task, where F_i is the set of tasks that cannot be performed before task i is completed. Inequalities (4) and (5) guarantee that the each workstation's station time cannot exceed the total processing time (t_{sum}) and the cycle time (c), respectively. The last set of the constraints (6) is the necessary condition for the existence of a line balance.

3. GRASP Method

Greedy randomized adaptive search procedure (GRASP) is firstly developed by Feo and Resende (1995) and is an iterative randomized sampling technique in which each iteration provides a solution to the problem at hand. GRASP is a constructive search method, it starts with an empty solution and adds solution elements to a partial solution until it is complete. Each GRASP iteration consists of two phases, a construction phase and a local search phase. In the first phase, an initial solution is constructed intelligently via an adaptive randomized greedy function. In the second phase, a local search procedure is implemented to the constructed solution in hope of finding an improvement. The best overall solution is kept as the result.

GRASP technique have been applied to a large variety of optimization problems such as scheduling (Feo et al. 1991; Ribeiro and Urrutia 2007), routing (Arguello et al. 1997; Boudia et al. 2007), partitioning (Arguello et al. 1996), and assembly line balancing (Bautista et al. 2000; Andres et al. 2008, Scholl et al. 2011). Bautista et al. (2000) introduced a GRASP technique which is obtained from the application of some classic heuristics, based on priority rules, and a genetic algorithm that searches for the solution in the heuristic space. They addressed the SALBP with the aim of minimizing the number of workstations and the cycle time for the minimum number of workstations.

Andres et al. (2008) applied GRASP in order to solve the SALBP with sequence-dependent setup times. Different heuristic rules and a GRASP algorithm compared in their research. Scholl et al. (2011) modified the SALBP with the sequence-dependent setup times more realistically and used GRASP.
Since the solution has to be constructed in the first phase of GRASP, tasks should be assigned step by step to the partial solution. Firstly, candidate tasks are ranked according to the candidate evaluation function, namely the greedy function (g_i), in such a way that the task with the best greedy index (see formula 7) is the first. After ordering the candidates, the restricted candidate list (RCL), a list of promising candidate elements, is established. From this list a random element is chosen and added to the current partial solution. Then, the greedy functions of the tasks are adapted and a new RCL is occurred. These steps are repeated until all tasks assigned.

In this study, we use the following greedy index (Andres et al. 2008):

$$Index_i^k = 1/t_i^k \tag{7}$$

where t_i^k is the operation time of task i in iteration k. The lower the index, the more interesting the task is to be sequenced in iteration k.

Let $g_{\min}^k$ and $g_{\max}^k$ are the minimum and maximum index values in iteration k. In order to form the RCL in iteration k (RCL_k) a threshold is defined so that all candidate tasks whose $Index_i^k$ is above the threshold will be selected for RCL_k..

$$Threshold^k = g_{\min}^k + \alpha \cdot \left(g_{\max}^k - g_{\min}^k\right) \tag{8}$$

The parameter $\alpha \in [0,1]$ controls the trade-off between randomness and greediness in the constructive process. Thus, a value $\alpha = 1$ means that all candidate tasks can be selected in iteration k. In such a case, the algorithm will be purely random. On the contrary, a value $\alpha = 0$ means that RCL is limited to 1, then only the task with the most positive impact will be selected. In that case, the algorithm will be purely greedy.

At the end of the construction phase a feasible task sequence is obtained. In order to improve the solution, neighborhood solutions are studied in the local search phase. An exchange movement that tries to exchange the positions of every two tasks in the sequence, provided that the exchange is feasible, is applied. For every feasible exchange, the variation in the objective function value is computed. If the variation is improvement, then it is accepted and the modified solution becomes the current solution. The local search ends when no feasible exchange can improve the current solution.

Since our real-life balancing problem is SALBP-2, we use GRASP through an iterated procedure that solves the corresponding SALBP-1 starting with a cycle time equal to the theoretical minimum and progressively being increased by an amount of time until an assignment of all the tasks to the m stations has been achieved. Pseudo code of the proposed GRASP algorithm is in the following:

Step 1: Set cycle time (c) equal to the lower bound (LB = $\max\{t_{max}, t_{sum}/m\}$) for the first iteration, and determine the number of workstations (m) aimed to be achieved.
Step 2: Determine the value of the parameter α, and maximum number of iterations (k_{max}).
Step 3: Determine candidates.
Step 4: Evaluate candidates via a greedy function g(x) (Andres et al. 2008).
$g(x) = 1/t_i$
Step 5: Form the RCL by using the threshold
$\mu = g_{min} + \alpha * (g_{max} - g_{min})$
Step 6: Choose a task from RCL randomly.
Step 7: Determine new candidates and adapt their greedy function $g(x)$.
Step 8: Repeat steps 3-7 untill an initial task sequence (TS) is obtained.
Step 9: Assign tasks to the workstations according to the TS by assigning as many possible tasks into the first m-1 workstations. Assign all the remaining tasks to the last workstation, m.

Step 10: Calculate the station time St_z ($z \in WS$) and potential station time ($PSt_z = St_z$ + the processing time of the first task assigned to (z+1)st station) of each station.
Step 11: Set $c_w = \max\{ St_1, St_2, \ldots, St_m\}$ and $c = \min\{ PSt_1, PSt_2, \ldots, PSt_m\}$.
Step 12: If $c_w > c$, then repeat steps 9-11.
Otherwise means that the predetermined m values is obtained, and go to step 13.
Step 13: Compute the fitness function of the *TS*.
Step 14: Determine the best fitness and its associated ALB solution.
if ($k = 0$) then ; $g_{best} = g_0$; $TS_{best} = TS_0$.
else if $g_k < g_{best}$ then $TS_{best} = TS_0$.
Step 15: Determine the feasible neighbors of *TS*; and repeat steps 13-15 until a local optimum is found.

Flow diagram of the proposed GRASP approach is depicted in Figure 1.

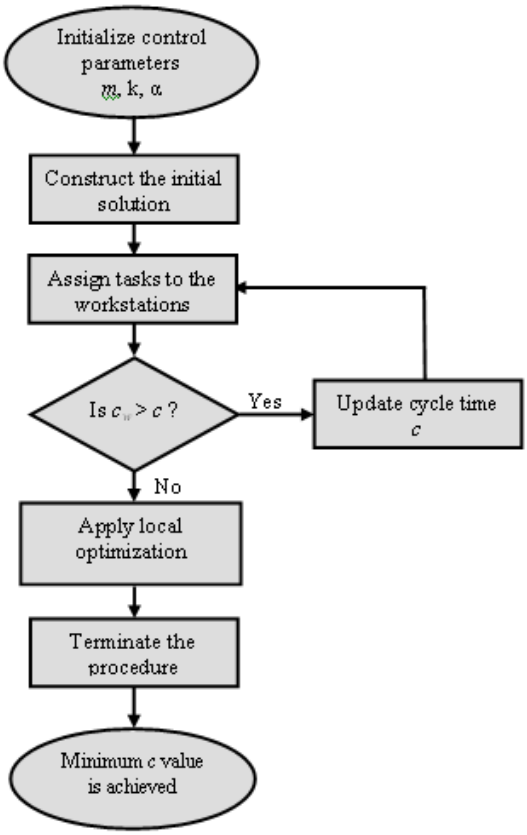

Figure 1: Flow diagram of the proposed GRASP algorithm

4. PSO Algorithm

One of the newest and most sophisticated meta-heuristics is particle swarm optimization (PSO) which is a population based meta-heuristic inspired by the social behavior of bird flocks and fish schools (swarms) searching by food and is introduced by Eberhart and Kennedy in 1995. Since that, PSO became a very popular topic for the researchers. It has been applied to several optimization problems such as scheduling (Liaoa et al. 2007; Liua et al. 2008), traveling salesman (Clerc 2004; Shi et al. 2007) and assignment problems (Yin et al. 2006).

PSO algorithm searches the feasible solution space (Ω) of the problem and updates the set of particles, namely the swarm, to find the global optimal solution. Let *Ns* be the number of particles, in other words the swarm size. Each particle of the swarm i ($i = Ns$) corresponds to a feasible solution of the problem and flies in the D-dimensional search space.

Each particle is updated at every iteration according to its best position and the best position of the whole swarm. Every single particle has the following attributes which consists of D-dimensional parameter vectors:

- its current position $x_{i,k}$,
- its personal best position achieved so far $x_{i,k}^{pbest}$,and
- its current velocity $x_{i,k}$.

The index k denotes the iteration number of the algorithm.
The proposed PSO algorithm is outlined step by step as follows:

Step 1: Set cycle time (c) equal to the lower bound (LB = max $\{t_{max}, t_{sum}/m\}$) for the first iteration, and determine the number of workstations (m) aimed to be achieved.
Step 2: Generate a random initial swarm $S = \{x_{1,0}, x_{2,0}, \ldots, x_{Ns,0}\}$ by using the following formulas.

$$x_{i,0}^{j} = x_{\min} + rand(x_{maz} - x_{\min}) \tag{9}$$

$$v_{i,0}^{j} = v_{\min} + rand(v_{\max} - v_{\min}) \tag{10}$$

where j denotes the dimension (j = 1, …, D); min and max values of x and v are user-defined values; and rand is a uniform random number in (0,1).
Step 3: Encode all individuals by using priority based encoding (Gen and Cheng 2000; Neachou 2011).
Step 4: Decode all particles as in above approach and evaluate the fitness function of their corresponding solutions.
Step 5: Determine each particle's own best position and the global best position of the whole swarm corresponding to the fitness function.
Step 6: Update the velocity of each particle as follows:

$$v_{i,k} = c_1 r_1 (x_{i,k}^{pbest} - x_{i,k}) + c_2 r_2 (x_{i,k}^{gbest} - x_{i,k}) + Iw_k v_{k-1} \tag{11}$$

$$Iw_k = \Theta Iw_{k-1} \tag{12}$$

where c_1 and c_2 are two parameters the cognitive and social parameters, respectively; r_1, r_2 are uniform random number in (0,1); and Iw_k is called inertia weight. If the values of c_1, c_2 and Iw_k are properly chosen, it is guaranteed that the particles' velocities do not grow to infinity (Clerc and Kennedy 2002). Θ is a constant that decreases the inertia weight at every iteration.
Step 7: Calculate all individuals' new position by:

$$x_{i,k} = x_{i,k-1} + v_{i,k} \tag{13}$$

Step 8: Repeat steps (3) – (7) until the termination criterion for the PSO is satisfied.
Step 9: If the aimed number of workstations is satisfied by the best particle stop, else update cycle time as in the following and go to Step 3.

Flow diagram of the proposed GRASP approach is depicted in Figure 2. The mentality of the proposed PSO algorithm to solve SALBP-2 is illustrated in the figure. In this approach, tasks are assigned to workstations according to priority based encoding. Each particle's position in the swarm determines the priority of the corresponding task, and a feasible task sequence is established according to the priorities.

5. Industrial Application

Electric unit production is a complicated process with many specific operations such as assembling transformer, serigraphy, quality control, etc. The assembly process of an electric unit in an electric plant is examined in this study. The precedence graph for the investigated assembly line is given in Figure 3. The numbers inside the nodes of the graph correspond to the tasks. According to the precedence relations, e.g., tasks 4, 6, and 7 must be completed before task 5 begins to operate, so 4, 6, 7 are direct predecessors of task 5. In addition, task 5 must be completed before task 9, because task 9 is a direct successor of task 5.

The operation times of tasks on the examined assembly line are given in the following table. According to the table task 1 has a task time of 47, task 2 has a task time of 29, and so on…

In order to apply the proposed meta-heuristics to the SALBP-2, the control parameters have to be selected at first. For GRASP, it is offered to use $\alpha = 0.3$ in the relevant literature (Andres et al. 2008). Also the maximum number of iterations is set as 10 because, there is no relative solution quality increase more than 10 iterations. For PSO, The

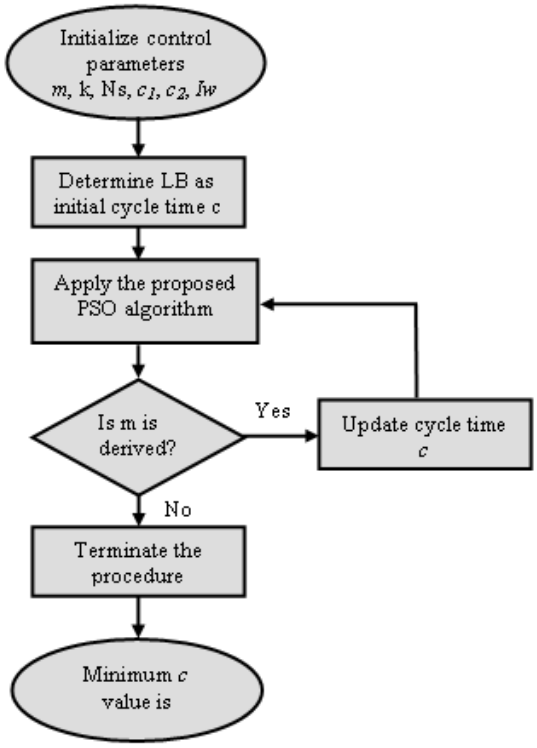

Figure 2: Flow diagram of the proposed PSO algorithm

swarm size (*Ns*) is considered as equal to the number of tasks, 18 for our problem. Maximum number of iterations is set equal to 100 times of the swarm size. Cognitive (c_1) and social (c_2) parameters are defined to be 2 and 1.5, respectively. Inertia weight is set equal to 0.9 and decreased by $\Theta = 0.95$.

In our industrial case study the number of workstations in the line is 7. The proposed meta-heuristics are implemented by using MATLAB version 2009b and the experimentation is carried out on a Pentium-IV 3.40 GHz PC.

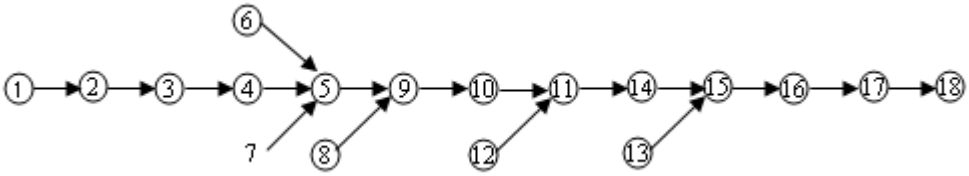

Figure 3: Precedence Relations of the proposed problem

Table 1. Processing times of tasks

Task	Processing time	Task	Processing time	Task	Processing time
1	47	7	31	13	7
2	29	8	0	14	21
3	76	9	40	15	44
4	24	10	42	16	45
5	25	11	53	17	14
6	23,5	12	34	18	100

As a result, the assignment of tasks to the workstations via GRASP and PSO algorithms are shown in Figures 4.a and 4.b, respectively. The figure illustrates the allocation of tasks among the workstations. The nodes denote the tasks and rectangles denote the workstations. According to the solution obtained by GRASP, tasks 8, 12, 13 and 1 are assigned to workstation 1; 2 and 3 are processed on workstation 2; 6, 4, 7 and 5 are operated on workstation 3; 9 and 10 are assigned to workstation 4; 11 and 14 are performed on workstation 5; 15, 16 and 17 are assembled on workstation 6; finally, task18 is processed on workstation 7. As a result, the minimum cycle time (*c*) value is computed as 105.

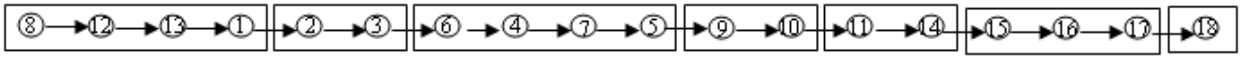

Figure 4(a): The rebalanced assembly line of the electric plant via GRASP

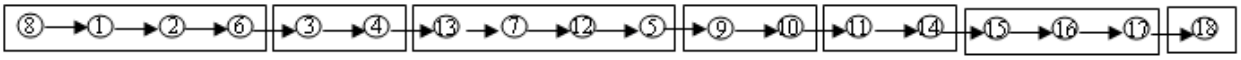

Figure 4(b): The rebalanced assembly line of the electric plant via PSO

As it can be seen from Figure 4.b, according to the PSO algorithm's solution, tasks 8, 1, 2 and 6 are assigned to workstation 1; 3 and 4 are processed on workstation 2; 13, 7, 12 and 5 are operated on workstation 3; 9 and 10 are assigned to workstation 4; 11 and 14 are performed on workstation 5; 15, 16 and 17 are assembled on workstation 6; finally, task18 is processed on workstation 7. As a result, the minimum cycle time (*c*) value is computed as 103.

Table 2. Results of the rebalanced assembly line

	Before Balancing	GRASP	PSO
Cycle time	131.5	105	103
Smoothness index	16.223	10.551	8.938

The results of the rebalanced assembly line are given in Table 2. Before balancing the assembly line, cycle time of the line was 131.5 minutes. After balancing the line by using GRASP and PSO the cycle time is decreased to 105 and 103 minutes, respectively which mean that the cycle time is reduced by 20.15% and 21.67%, respectively. Moreover, the workload is more smoothened with the use of meta-heuristics. Another conclusion can be made from table 2 that PSO outperforms GRASP for this industrial case problem.

6. Conclusions

Assembly line rebalancing is an important aspect in the industrial environment because of the dynamic nature of the production processes. In such cases, when an existing line has to be rebalanced assembly line balancing problem of type-2 occurs.

In this paper a real-life assembly line balancing problem of type 2 is addressed. Two meta-heuristics: greedy randomized adaptive search procedure (GRASP) and particle swarm optimization (PSO) are introduced for the bi-objective SALBP-2. The proposed algorithms are explained in detail. The same methodology is applied to the mentioned problem with these two meta-heuristics. As a result, both GRASP and PSO have a meaningful improvement in both of the objectives. However, it is concluded that PSO outperforms GRASP on both cycle time and workload smoothness objectives for the studied problem.

Further research will focus on the practical application of the proposed heuristics to solve more complex assembly line balancing problems.

References

Andres, C., Miralles, C., and Pastor, R., 2008, "Balancing And Scheduling Tasks in Assembly Lines With Sequence-Dependent Setup Times", European Journal of Operational Research, 187, 1212–1223.

Arguello, M.F., Feo, T.A., and Goldschmidt, O., 1996, “Randomized methods for the number partitioning problem”, Computers and Operations Research, 230,103–111.

Arguello, MF., Bard, J.F., and Yu, G., 1997, “A GRASP for Aircraft Routing in Response to Groundings and Delays”, Journal of Combinatorial Optimization, 1, 211–228.

Bautista, J., Sudrez, R., Mateo, M., and Companys, R., 2000, “Local Search Heuristics for the Assembly Line Balancing Problem with Incompatibilities Between Tasks”, Proceedings of the 2000 IEEE International Conference on Robotics and Automation, 2404-2409.

Boudia, M., Louly M.A.O., and Prins, C., 2007, “A Reactive GRASP and Path Relinking for a Combined Production-Distribution Problem”, Computers and Operations Research, 34, 3402–3419.

Clerc, M., and Kennedy, J., 2002, “The Particle Swarm-Explosion, Stability and Convergence in a Multidimensional Complex Space” IEEE Transactions on Evolutionary Computation, 6, 58-73.

Clerc, M., 2004, “Discrete Particle Swarm Optimization, Illustrated by the Traveling Salesman Problem”, in New Optimization Techniques in Engineering, 219-239.

Eberhart, R.C., and Kennedy, J., 1995, “A New Optimizer Using Particle Swarm Theory”, Proceedings of International Symposium on Micro Machine and Human Science, 39-43.

Feo, T.A., Venkatraman, K., and Bard, J.F., 1991, “A GRASP for a Difficult Single Machine Scheduling Problem”, Computers and Operations Research, 18, 635–643.

Feo, T.A., Resende M.G.C., 1995, “Greedy Randomized Adaptive Search Procedure”, J. of Global Optimization, 6, 109–133.

Gen, M., and Cheng, R., 2000, Genetic Algorithms and Engineering Optimization, 1st Edition, Wiley-Interscience, New York.

Hackman, S.T., Magazine, M.J., and Wee, T.S., 1988, “Fast, Effective Algorithms for Simple Assembly Line Balancing Problems”, Operations Research, 37, 916-924.

Karp, R.M., 1972, Reducibility among Combinatorial Problems, Miller R.E and Thatcher J.W. (eds.) Complexity of Computer Applications, Plenum Press, New York, 85-104.

Klein, R., Scholl, A., 1996, “Maximizing the Production Rate in Simple Assembly Line Balancing-A Branch and Bound Procedure”, European Journal of Operational Research, 91, 2 367-385.

Liaoa, C.J., Tsengb, C.T., and Luarn, P., 2007, “A Discrete Version of Particle Swarm Optimization for Flowshop Scheduling Problems”, Computers and Operations Research, 34, 3099-3111.

Liua, B., Wanga, L., and Jina, Y.H., 2008, “An Effective Hybrid PSO-Based Algorithm for Flow Shop Scheduling with Limited Buffers”, Computers and Operations Research, 35, 2791-2806.

Nearchou, A.C., 2007, “Multi-Objective Balancing of Assembly Lines By Population Heuristics”, International Journal of Production Research, 46, 2275-2297.

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.

Ribeiro, C.C., and Urrutia, S., 2007, “Heuristics for the Mirrored Traveling Tournament Problem”, European Journal of Operational Research, 179, 775–787.

Salveson, M.E., 1955, “The Assembly Line Balancing Problem”, Journal of Industrial Engineering, 6, 18-25.

Seyed-Alaghebanda, S.A., Fatemi Ghomia, S.M.T., and 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.

Scholl, A., and Voβ, S., 1996, “Simple Assembly Line Balancing-Heuristic Approaches”, Journal of Heuristics, 2, 217-244.

Scholl, A., 1999, Balancing and Sequencing of Assembly Lines, 2nd Edition, Physica-Verlag, Heidelberg.

Scholl, A., Becker, C., 2006, “State-Of-The-Art Exact and Heuristic Solution Procedures for Simple Assembly Line Balancing”, European Journal of Operations Research, 168, 666-693.

Scholl, A., Boysen, N., and Fliedner, M., 2011, “The Assembly Line Balancing and Scheduling Problem with Sequence-Dependent Setup Times: Problem Extension, Model Formulation and Efficient Heuristics”, OR Spectrum.

Shi, X.H., Liang, Y.C., Lee, H.P., Lu, C., and Wang, Q.X., 2007, “Particle Swarm Optimization-Based Algorithms for TSP and Generalized TSP”, Information Processing Letters, 103, 169-176.

Simaria, A.S., and Vilarinho, P.M., 2004, “A Genetic Algorithm Based Approach to the Mixed-Model Assembly Line Balancing Problem of Type II”, Computers and Industrial Engineering, 47, 391-407.

Yin, P.Y., Yu, S.S., Wang, P.P., and Wang, Y.T., 2006, “A Hybrid Particle Swarm Optimization Algorithm for Optimal Task Assignment in Distributed Systems”, Computer Standards and Interfaces, 28, 441-450.