Simple Assembly Line Balancing Using Particle Swarm Optimization Algorithm
Qi Lv

Simple Assembly Line Balancing Using Particle Swarm Optimization Algorithm
Qi Lv
North China University of Water Resources and Electric Power, School of Management and Economics, Zhengzhou, China, lv.qi@foxmail.com
doi:10.4156/jdcta.vol5.issue6.36

Abstract
This paper presents the application of particle swarm optimization (PSO) algorithm for the simple assembly line balancing problem, SALBP-I. A new indirect encoding method for the solution of SALBP-I is developed to keep the feasibility of operation sequence. The particle that represents a feasible operation sequence (FOS) is based on a smallest position value rule. Given the FOS defined by a particle, the optimal assignment of the operations to the workstations is identified by an optimum-seeking procedure with polynomial-time complexity. Then PSO is employed to find the optimum efficiently in the search space comprising the optimal assignments associated with all FOSs. The PSO algorithm is tested on a set of problems taken from the literature and compared with other approaches. The computation results show the effectiveness of the algorithm.

Keywords: Simple Assembly Line Balancing, Particle Swarm Optimization, Operation Sequence, Smallest Position Value

1. Introduction
Assembly lines are flow-oriented production systems which are adopted typically in the industrial production of high quantity standardized commodities and even become important in low volume production of customized products [1]. An assembly line consists of a sequence of m workstations, usually connected by a conveyor belt, through which the product units flow (see Figure 1). Each workstation performs a subset of the n operations necessary for manufacturing the products. Each product unit remains at each station for a fixed time called the cycle time, C.

For the decision problems arising in managing assembly lines, assembly line balancing problems are important tasks in medium-term production planning. The decision problem of optimally partitioning the assembly works among the stations to achieve some objective is known as the assembly line balancing problem (ALBP). To learn more on classification of ALBPs, one can refer to [2]. Any type of ALBP deals with finding a feasible line balance, i.e., an assignment of each task to exactly one station such that the precedence constraints and possibly further restrictions are fulfilled.

Among ALBPs, a simple problem consists of assigning operations to workstations such that the line efficiency is maximized or the number of workstations is minimized, which refers to simple assembly line balancing problem of type 1, i.e., SALBP-I. Since SALBP-I can be viewed as the extension of the well-known bin packing problem, Wee and Magazine [1] showed that SALBP-I is an NP-hard problem. Due to the NP-hard essence of SALBP-I, many heuristics except for exact methods have been developed for the problem since 1950’s (see [1] for details). As for meta-heuristics for SALBP-I, Scholl [1] identified and reviewed the genetic algorithms (GAs), tabu search (TS) procedures, simulated annealing (SA) procedures and ant colony optimization (ACO) approach. However, the relatively new population-based meta-heuristic particle swarm optimization (PSO) algorithm is seldom applied for SALBP-I in literature [4].

In this paper, we attempt to use particle swarm optimization algorithm for SALBP-I. Moreover, a new indirect encoding method for the solution of SALBP-I is developed. To reduce the complexity of search space and improve the efficiency of the algorithm, the PSO is employed to find the optimum in the refined search space which comprises the optimal assignments associated with all feasible operation sequences (FOSs). The comparisons between PSO and other approaches show the effectiveness of the PSO algorithm.

The remainder of the paper is organized as follows: in Section 2, the SALBP-I problem is stated in
detail. The PSO optimization approach is described in Section 3. In Section 4, a set of problems taken
from the literature are used to illustrate the effectiveness and efficiency of the approach. A conclusion
is given in Section 5.

Figure 1. Example of an assembly line

2. Problem Statement

Most of the research in assembly line balancing has been devoted to modelling and solving the
simple assembly line balancing problem (SALBP).

2.1. Problem definition

Baybars [5] specifies the following assumptions for the SALBP:

- all input parameters are known with certainty;
- a task (operation) cannot be split among two or more stations;
- tasks cannot be processed in arbitrary sequences due to technological precedence requirements;
- all tasks must be processed;
- all stations under consideration are equipped and manned to process any task;
- task times are independent of the station at which they are performed and of the preceding
tasks;
- any task can be processed at any station;
- the line is serial;
- the line is designed for a unique model of a single product.

The SALBP-I as a typical version of SALBP is to find the optimal assignments of operations in
order to minimize the number of workstations, satisfying the precedence constraints and cycle time
constraint. The precedence relations of operations (tasks) are usually represented by a precedence
graph (PG). It contains a node for each task, node weights for the task times and arcs for the
precedence constraints. Figure 2 shows a precedence graph with n = 7 operations having task times
between 1 and 6 (time units). In this paper, an operation sequence satisfying the precedence relations
represented by a PG is named feasible operation sequence (FOS). It is clear that a feasible assignment
of operations along an assembly line must be a FOS.

Figure 2. Precedence graph with 7 operations
2.2. Problem formulation

Nomenclature:
- $N$ number of operations (tasks),
- $k$ workstation number,
- $t_i$ time of the task $i$ (deterministically known),
- $C$ line cycle time,
- $P_i$ set of immediate predecessors of the task $i$.

Using some weights (cost) of the station assignment $w_{ik} = w_k$ and $N \cdot w_k \leq w_{ik}, \forall k$, SALBP-I can be presented as the following integer programming problem [6]:

Min:
$$\sum_{i=1}^{N} \sum_{k=1}^{N} w_{ik} x_{ik}$$

Subject to:
$$\sum_{k=1}^{N} x_{ik} = 1, \ \forall i = 1, 2, \ldots, N$$

$$\sum_{i=1}^{N} t_i x_{ik} \leq C, \ \forall k = 1, 2, \ldots, N$$

$$x_{ik} \leq \sum_{j=1}^{N} x_{ij}, \ \forall i, k = 1, 2, \ldots, N \text{ and } i \in P_i$$

$$x_{ik} \in \{0, 1\}$$

where $x_{ik}$ equals 1 if operation (task) $i$ is assigned to workstation $k$, otherwise $x_{ik}$ equals zero.

From the above formulation, it can be seen that SALBP-I reduces to the well known bin-packing problem if precedence constraints are omitted. Since SALBP-I is more complex than the bin-packing problem which is known to be NP-hard, some meta-heuristics such as GAs and SA algorithms are used to solve the problems. In this paper, a relatively new meta-heuristic, namely PSO, is applied to solve the SALBP-I.

3. Particle Swarm Optimization Approach

3.1. Introduction to PSO

Particle swarm optimization (PSO) was developed by Kennedy and Eberhart [7]. The PSO is inspired by the social behavior of a flock of migrating birds trying to reach an unknown destination. In PSO, each solution is a ‘bird’ in the flock and is referred to as a ‘particle’. A particle is analogous to a chromosome (population member) in GAs. As opposed to GAs, the evolutionary process in the PSO does not create new birds from parent ones. Rather, the birds in the population only evolve their social behavior and accordingly their movement towards a destination [8].

The evolution process of PSO is initialized with a group of random particles (solutions). The $i$th particle is represented by its position as a point in a $S$-dimensional space, where $S$ is the number of variables. Throughout the process, each particle $i$ monitors three values: its current position ($X_i$); the best position it reached in previous cycles ($P_i$); its flying velocity ($V_i$). In each time interval (cycle), the position ($P_g$) of the best particle $g$ is calculated as the best fitness of all particles. Accordingly, each particle updates its velocity $V_{i+1}$ and position $X_{i+1}$ to catch up with the best particle $g$, as follows:

$$V_{i+1} = \omega V_i + C_1 \text{rand} (P_i - X_i) + C_2 \text{rand} (P_g - X_i)$$

$$X_{i+1} = X_i + V_{i+1}, \quad V_{\text{max}} < V_{i+1} < V_{\text{max}}$$

where $\omega$, $C_1$, and $C_2$ are constants.
where C1 and C2 are two positive constants, namely, acceleration coefficients, rand() and Rand() are two random functions in the range [0, 1], $V_{\max}$ is an upper limit on the maximum change of particle velocity, and $\omega$ is an inertia weight employed as an improvement proposed by Shi and Eberhart [8] to control the impact of the previous history of velocities on the current velocity. The operator $\omega$ plays the role of balancing the global search and the local search; and was proposed to decrease linearly with time [8]. The pseudocode for a PSO procedure is as follows:

Algorithm PSO
Begin
    Generate random population of N solutions (particles);
    For each individual $i \in N$ calculate fitness ($f_i$);
    Initialize the value of the weight factor $\omega$;
    For each particle;
        Set $pBest$ as the best position of particle $i$;
        If fitness ($f_i$) is better than $pBest$;
            $pBest(i)$ = fitness ($f_i$);
        End;
        Set $gBest$ as the best fitness of all particles;
        For each particle;
            Calculate particle velocity according to Eq. (6a);
            Update particle position according to Eq. (6b);
        End;
        Update the value of the weight factor $\omega$ (option);
    End;
    Check if termination = true;
End

The above PSO is for continuous optimization problem and cannot be used directly for discrete optimization problem such as the SALBP-I. To apply PSO to SALBP-I, suitable solution representation and particle’s position updating method need to be developed.

3.2. Solution representation

For SALBP-I, two solution encoding schemes are usually used: direct encoding and indirect encoding [1, 9]. For the direct encoding scheme, the encoding records the full information of the solution including the task ID and corresponding ID of assigned workstation for every task. On the contrary, the indirect encoding scheme only records part information for the solution of SALBP-I such as the task (operation) sequence. One merit of indirect encoding is the easiness of keeping the feasibility of each encoding. To keep the feasibility of each particle which represents a potential solution, we adopt the indirect encoding scheme.

A particle only records the FOS instead of the whole solution and an optimum-seeking algorithm is used to find the optimal solution given a FOS. Based on our previous work [10] for representing a FOA by a priority based encoding method, a smallest position value (SPV) method [11] is utilized to record a permutation of $N$ numbers for each particle in the PSO algorithm. Figure 3 shows the representation of a FOS in the PG represented by Figure 2. The characteristic of SPV is to represent a permutation of $N$ numbers by $N$ real numbers, and then Eq.(6) can be used to change velocity and position of particle $i$.

For the SPV, let $V_{\max}=N-1$ and $-V_{\max}=-(N-1)$.

Figure 3. Particle representation for a permutation of $N$ numbers
To obtain a FOS by the SPV method, firstly a permutation of N numbers is constructed by SPV rule \[12\] in light of N real numbers as shown in Figure 3 (see \[11\] for details); then a zero in-degree topological sorting technique \[10\] is used to get the FOS in light of the relating PG and the permutation of N numbers. The permutation of 7 numbers and the FOS corresponding to Figure 3 are 2-1-4-5-7-6-3 and 1-4-7-2-5-6-3 respectively.

3.3. Optimum seeking procedure for a FOS

As aforementioned, an indirect encoding scheme is used to represent the solution of SALBP-I, which only records the information of a FOS. To get the full information of a solution, the optimal partition of the FOS should be determined. Here, an efficient optimum寻求 procedure proposed by Klein \[13\] is adopted to identify the optimal partition of the FOS. Assuming that the order of a FOS under consideration is (1,2, ..., n), Klein \[13\] proved that the following procedure minimizes the number of work stations considering an arbitrary assignment of all operations to m stations:

\[
\text{Algorithm optimum寻求} \\
\begin{align*}
\text{Begin} \\
\text{Repeat} \\
\quad \text{Assign the first } s \text{ operations to the first station, where } s \text{ satisfies } \sum_{i=1}^{s} t_i \leq C < \sum_{i=1}^{s+1} t_i; \\
\quad \text{Until all operations have been assigned;} \\
\text{End}
\end{align*}
\]

It is clear that the complexity of the above algorithm is \(O(N)\) for N operations. Since the optimum寻求 algorithm is a strong polynomial-time algorithm, it must be high efficient.

3.4. PSO optimization procedure

During the calculation of fitness for each particle, the following fitness function \[14\] for SALBP-I is adopted:

\[
f = \sqrt{\frac{\sum_{k=1}^{n} (t_{\text{max}} - t_k)^2}{n}} + \frac{\sum_{k=1}^{n} (t_{\text{max}} - t_k)}{n} \tag{7}
\]

where \(n\) is the number of stations, \(t_{\text{max}}\) is the maximum station time, and \(t_k\) is the \(k\)th station time. The first part of Eq.(7) aims to find the best balance among the solutions that have the same number of stations while the second part minimizes the number of stations in the solution. Here, we assume that the first objective is the same critical with the second one.

As for particle’s position updating method, Eq. (6) is adopted because each element of the particle is a real number with the SPV encoding method. After updating a particle’s position, the particle is still feasible, which can be decoded to a permutation of \(N\) numbers and a FOS.

The procedure of the PSO algorithm for the SALBP-I is the same with the procedure in Subsection A. In this paper, if the PSO algorithm has reached MAXGEN generations in a run, the procedure ends. Thus the parameters of the PSO algorithm comprises maximum generations of a run MAXGEN, the number of particles of a population POPSIZE, acceleration coefficients C1 and C2 as well as inertia weight \(\omega\).

According to above description, the PSO is employed to maintain a swarm of FOSs and identify optimal solution through randomly searching in the refined search space which is composed of the optimal partitions associated with all FOSs. Usually the proportion of FOSs to all operation sequences is quite small \[10\], therefore the refined search space is much smaller than the whole search space. Through the development of suitable encoding methods and relating position updating mechanism, the search space is limited to the refined search space. Searching in this space improves the efficiency of the PSO and increases the possibility to find the optimum. Figure 4 depicts the whole idea of the PSO optimization approach.
4. Computational Results

To demonstrate the effectiveness and robustness of the PSO optimization approach, we present computational results obtained on a set of SALBP-I problems selected in the literature\cite{15,16}.

4.1. Description of test problems

Four classes of instances for SALBP-I are selected. The basic characteristics of the selected problems are summarized in Table 1. The first column displays the author of the instance. The second column shows the number of tasks. The third and fourth column shows the minimum and maximum cycle time. The remaining columns contain the minimum processing time, the maximum processing time, the sum of processing times, the order strength (OR) in percent.

<table>
<thead>
<tr>
<th>Author</th>
<th>n</th>
<th>Cmin</th>
<th>Cmax</th>
<th>tmin</th>
<th>tmax</th>
<th>tsum</th>
<th>OR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mertens</td>
<td>7</td>
<td>6</td>
<td>18</td>
<td>1</td>
<td>6</td>
<td>29</td>
<td>52.4</td>
</tr>
<tr>
<td>Mitchell</td>
<td>21</td>
<td>14</td>
<td>39</td>
<td>1</td>
<td>13</td>
<td>105</td>
<td>71</td>
</tr>
<tr>
<td>Sawyer</td>
<td>30</td>
<td>25</td>
<td>75</td>
<td>1</td>
<td>24</td>
<td>324</td>
<td>44.8</td>
</tr>
<tr>
<td>Tonge</td>
<td>70</td>
<td>160</td>
<td>527</td>
<td>1</td>
<td>156</td>
<td>3510</td>
<td>59.4</td>
</tr>
</tbody>
</table>

The tests were run on a personal computer with a Pentium-IV 1.7GHz CPU and 512 MB memory. The algorithm was coded in Visual C++ 6.0. The parameters of the PSO algorithm are determined by recommended values and trial and error. The PSO algorithm had the following configuration: POPSIZE=30, MAXGEN=500, C1=C2=2.0, \(\omega=0.8\).

4.2. Computational results

The above parameters were held constant for all problems. The results of experimental tests for four classes of instances are presented in Table 2. We use the data set of [15] and compared the results of the proposed PSO algorithm with the ones obtained by EUREKA provided by [16]. As can be observed in Table 4, the proposed PSO algorithm produces solutions that are as good as the EUREKA heuristic of Hoffman. For all the instances of selected problems, the PSO algorithm is able to find the optimum quickly. Comparing the generation no of best solution between the PSO and GA in [16], we find that the PSO obtains the optimum faster than the GA when the number of operations are 30 and 70. In [16] the POPSIZE is 70 which is bigger than the used POPSIZE (30) in the PSO algorithm. This may indicate that the PSO converges to the optimum quicker than the GA in [16]. The comparison results show that the PSO is effective and efficient for SALBP-I.
5. Conclusion

A particle swarm optimization (PSO) algorithm for the simple assembly line balancing problem, SALBP-I, is presented. A new indirect encoding method for the SALBP-I is developed, which records the FOS using a smallest position value (SPV) method. Given a FOS represented by a particle, an optimum-seeking procedure with strong polynomial-time complexity is able to efficiently find the optimal solution associated with the FOS. Based on encoding method and position updating mechanism, the search space is narrowed to the optimal solution associated with all FOSs. Searching in the refined search space improves the efficiency of the PSO and increases the possibility to find the global optimum. The computational results of the proposed algorithm for a set of problems taken from the literature and comparisons between PSO algorithm and existing algorithms show that the proposed algorithm is effective and efficient. Further study is needed to compare the performance of PSO and other meta-heuristics such as GA and ACO in details. Multi-objective optimization for SALBP-I needs to be further investigated using PSO.

6. Acknowledgment

This work is supported by the fund for new staff from southeast university under grant No. 9202000023 and the open fund of Key Laboratory of Measurement and Control of Complex Systems of Engineering, Ministry of Education (No. 2010B002).

7. References