Balancing U-Shaped Assembly Lines with Resource Dependent Task Times: A Simulated Annealing Approach

Sachin Jayaswal[a,*], Prashant Agarwal[1]

[a]*Indian Institute of Management, Vastrapur, Ahmedabad, Gujarat 380 015, India. Ph: +91-79-6632-4877, Fax: +91-79-6632-6896, E-mail: sachin@iimahd.ernet.in*
[b]*Department of Industrial Engineering and Management, Indian Institute of Technology, Kharagpur, West Bengal 721302, India. E-mail: prashant.agarwal@iitkgp.ac.in*

Abstract

The advent of Just-in-Time (JIT) and Group Technology philosophies has popularized U-shaped assembly lines, which help overcome many of the disadvantages, like line inflexibility, job monotony, large inventories, etc., typically associated with straight assembly lines. Although U-shaped layout has demonstrated it supremacy over the traditional straight layout, the problem of U-shaped assembly line balancing (ULB) is much more complex. The extant literature on ULB assumes that each assembly task requires a fixed (or no) equipment and a fixed number of workers. However, it is often desirable to reduce certain task times by assigning more workers or alternative equipments at a given workstation. The problem in such cases is to assign not only the task but also resource alternatives (number of workers and equipment type) to workstations. Research on such resource dependent U-shaped assembly line balancing (RDULB) is scarce. We address the problem of RDULB and propose a Simulated Annealing (SA) based metaheuristic, which gives optimal solution for most of the small-to-medium problem instances. For very large problems, while SA generates a good feasible solution within half an hour to 1.5 hours, Cplex is unable to find a single feasible solution even after 10 times the CPU time required by SA.

Keywords: Assembly Line Balancing; Resource; Optimization; Simulated Annealing

*Corresponding author

1. Introduction & Literature Review

An assembly line is a sequence of workstations at which tasks related to assembly of a product are performed. Assembly line balancing (ALB), in the simplest sense, involves optimally (with respect to some objective) partitioning a set of tasks comprising the assembly of a product to these workstations, subject to the precedence relations among the tasks [2]. A common objective for assembly line balancing is to minimize the number of workstations for a given cycle time of assembly (Type-I problem). For other objectives and classifications of ALB, we refer the reader to Scholl [24].

Traditionally, the workstations on an assembly line are arranged in a straight line along a conveyor belt, and thus the assembly line is called a serial/straight line. However, the advent of Just-in-Time (JIT) and Group Technology philosophies has popularized U-shaped assembly lines, which help overcome many of the disadvantages, like line inflexibility, job monotony, large inventories etc., typically associated with straight assembly lines.

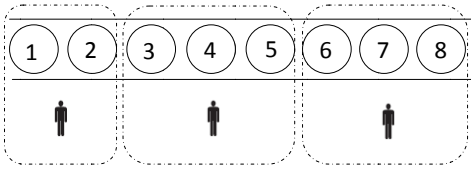

Figure 1: Straight layout

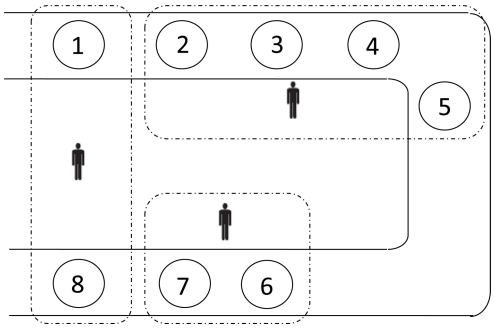

Figure 2: U-shaped layout

Figures 1 and 2 depict, respectively, the typical layout of a straight assembly line and that of a corresponding U-shaped line. The entrance and exit of U-shaped assembly lines are close together forming an 'U', and operators work inside this 'U'. This allows an operator to handle two workpieces, one on each side of the line, in the same cycle. Thus, a station k can contain not only tasks whose predecessors are assigned to one of the stations $1, ..., k$, but also tasks whose predecessors will be processed by the time the workpiece returns to station k for the second time [20]. Such a workstation that handles two workpieces in the same cycle (and equivalently the same workpiece in two different cycles) is called a crossover station (for example, the leftmost workstation in Figure 2 that processes tasks 1 and 8). The presence of such crossover stations usually increases the feasible task-workstation combinations. For example, in the U-shaped layout in Figure 2, task 8 can be assigned to the leftmost

workstation, which is not otherwise possible in a straight layout. This greater flexibility in task-workstation combinations often results in a better balancing of the line and fewer workstation (and hence operator) requirement. Closer workstations also increase the visibility of the whole production process and communications among operators. Operators thus get cross-trained on tasks of different workstations, which leads to their job enrichment and increased flexibility. This increased flexibility permits ease of frequent rebalancing of the line to quickly respond to changes in market demand/operating environment [24]. Other benefits associated with U-shaped assembly lines include: productivity improvement, reduction in work-in-process inventory, space requirement, and lead-time. Such benefits have been documented in various studies of U-shaped assembly lines [18, 19, 5, 1].

Although U-shaped layout has demonstrated its supremacy over the traditional straight layout, the problem of balancing the line in the former case is much more complex due to greater feasible station-task combinations. The problem of U-shaped line balancing (ULB) was first studied by Miltenburg and Wijngaard [16]. They developed a Dynamic Programming (DP) formulation to minimize the number of workstations for a given cycle time (Type-I problem), which could solve problems with up to 11 tasks. For problems of larger size, they proposed a Ranked Positional Weight Technique (RPWT) based greedy heuristic. Subsequently, Miltenburg and Sparling [17] developed three exact algorithms for the ULB problem: one DP based and two Branch & Bound (B&B) based. Urban (1998) developed the first Integer Programming (IP) based formulation of the ULB problem, using the concept of phantom precedence diagram. His formulation allowed assignment of tasks to stations forward on the original precedence diagram, and backward on the phantom precedence diagram. This could solve, using Cplex, problems of up to 45 tasks and 14 workstations, which could not earlier be solved by DP or B&B based algorithms of Miltenburg and Sparling [17]. Scholl and Klein [25] proposed a B&B based procedure, called ULINO (U-Line Optimizer), with various bounding and dominance rules to solve different versions (Type-I, Type-II and Type-E) of the ULB problem. Erel *et al.* [9] presented a Simulated Annealing (SA) based metaheuristic for solving large size ULB problems. Gokcen *et al.* [12] proposed a shortest route formulation of ULB. Gokcen and Agpak [11] and Toklu and Ozcan [26] developed Goal Programming formulations of ULB.

All the above cited papers on ULB assume that each assembly task requires a fixed (or no) equipment and a fixed number of workers. In such a case, the assembly line balancing problem is essentially that of partitioning a given set of tasks to different workstations.

Further, the objective of minimizing the number of workstations for a given cycle time (Type-I problem) automatically minimizes the total cost. However, it is often desirable (or even necessary to meet the cycle time requirement) to reduce certain task times by assigning more workers or alternative equipment at a given workstation. The problem in such cases is to assign not only the tasks but also resource alternatives (number of workers and equipment type) to workstations. Faaland *et al.* [10] called such a problem as resource dependent assembly line balancing (RDALB). The objective of minimizing the number of workstations for such problems may not necessarily result in the minimum total cost. Hence, for RDALB, the objective is to minimize the total cost, given the cycle time.

Most ALB problems in industries are actually RDALB [14]. For example, Faaland *et al.* [10] report the design of a JIT line to produce subassemblies for cruise missiles as an RDALB problem, in which case the managers had to decide not only the tasks but also the type of wire harness and test equipment at workstations. Kara *et al.* [14] report another instance of RDALB problem at a solar collector producer, located in Konya, Turkey. In this case, a task called 'glass positioning' could be performed by a worker in 16 seconds with the help of an assistant. The same task could be performed by that worker in 9 seconds with the help of a vacuum gripper.

Abundance of RDALB problems in industries has also motivated several theoretical research work. Pinto *et al.* [23] consider a closely related problem of simultaneous assignment of tasks to stations and choice of process alternatives. They propose a B&B procedure, which branches by selecting a processing alternative and computes lower and upper bounds by solving the resulting ALB problem. The effectiveness of the procedure is demonstrated by its application to redesigning of an assembly line for an auto-industry supplier. Bukchin and Tzur [3] also present a B&B procedure, which is capable of solving problems consisting of up to 30 tasks and 10 equipment types. For problems of a larger size, they propose a heuristic. Nicosia *et al.* [22] propose a DP and a B&B based solution procedure for the same problem. Corominas et al. (2008) present the problem of line rebalancing at a motorcycle assembly plant to increase its production by hiring the minimum number of temporary workers. Temporary workers need more time to carry out their tasks than permanent workers, and can work only with other permanent workers. Further, different task groups are incompatible with each other. The authors model the problem as a binary linear program (BLP), which is solved optimally by means of the Ilog Cplex 9.0 optimizer. Moon *et al.* [21] address the problem of assigning tasks to workstations and simultaneously

selecting multi-functional workers with different salaries depending on their skills. They propose a mixed integer linear program model and a genetic algorithm to minimize the total annual work station costs and the annual salary of the assigned workers.

All the above cited papers on RDALB apply to a serial layout. Research on resource dependent U-shaped assembly line balancing (RDULB) is very scarce. Kara *et al.* [14], to the best of our knowledge, is the only work on RDULB. They propose an integer program formulation, which considers alternative equipments and assistants to help regular workers with their tasks or to reduce their task times. The objective of the problem is to simultaneously assign tasks to workstations, and resources (equipments and assistants) to tasks such that the total cost of workstation and resource utilization is minimized, given the cycle time, precedence and resource restrictions. However, the resulting formulation has difficulty solving some problem instances even of the size of 30 tasks and 9 equipment types. This has motivated the current research to: (i) provide a tighter formulation of the problem; and (ii) develop an efficient algorithm capable of solving large instances of RDULB in reasonable time.

The rest of the paper is organized as follows: Section 2 presents the problem description and its mathematical model, followed by some ways to tighten it. The solution algorithm is discussed in detail in Section 3, followed by and illustrative example in Section 4 and computational experience in Section 5. The paper concludes with a summary of results and discussion of directions for future research in Section 6.

2. Problem Formulation

The problem description is adopted from Kara *et al.* [14]. The RDULB problem we consider in this paper has the following features:

- A single product is assembled on the assembly line.
- The precedence relationships among the tasks are given and fixed.
- Every task necessarily needs an operator, and some cannot be processed without additional resources (equipment or assistant). Others may be processed with or without additional resources.
- Task durations are deterministic, but vary with the type of resource (equipment type or assistant) used. Task durations are likely to be shorter when they use additional resources.

- Task durations are independent of the workstation at which they are performed.
- The equipments are bulky, and hence cannot be moved from one side (front/back) to the other (back/front) of a crossover workstation. However, regular operators and assistants can move, and hence can be utilized by different tasks on either side of a crossover workstation.
- Regular operators are available in sufficient number to operate the workstations but other resources (equipments and assistants) are limited.

We define the following notations, used to to develop the mathematical model of the problem.

Indices:

i, r, s	:	task
j	:	workstation
e	:	equipment type; $e = 0$ represents no equipment.

Sets:

T	:	Set of all tasks
W	:	Set of all workstations
E	:	Set of all equipment types
E_i	:	Set of equipment types that can be used to process task i
PR	:	Set of precedence relationships: $(r, s) \in PR$, where r is an immediate predecessor of s

Parameters:

NT : Number of tasks; $NT = |T|$

NW : Number of available workstations; $NW = |W|$

NE : Number of equipment types; $NE = |E|$

NE_e : Number of equipments of type e available

NA : Number of assistants available

CT : Cycle time

t_{ie0} : Duration of task i if it is processed on equipment e without an assistant

t_{ie1} : Duration of task i if it is processed on equipment e with the help of an assistant

cw : Annual utilization cost of a workstation (regular operator + amortized investment costs)

ca : Annual employment cost of an assistant

c_e : Annual operating cost of equipment e

M : a big number

Variables:

x_{ij} : 1, if task i is assigned to the front of workstation j; 0 otherwise

y_{ij} : 1, if task i is assigned to the back of workstation j; 0 otherwise

p_{ije} : 1, if task i is processed at workstation j using equipment e without an assistant; 0, otherwise

q_{ije} : 1, if task i is processed at workstation j using equipment e and with the help of an assistant; 0 otherwise

zf_{je} : 1, if equipment e is assigned to the front of workstation j; 0, otherwise

zb_{je} : 1, if equipment e is assigned to the back of workstation j; 0, otherwise

u_j : 1, if workstation j is utilized; 0, otherwise

k_j : 1, if an assistant is assigned to workstation j; 0, otherwise

Using the above notations, the mathematical model for RDULB, as proposed by Kara *et al.* [14], can be stated as:

$[RDULB]$:

$$\min \quad \sum_{j\in W}(cw.u_j + ca.k_j) + \sum_{e\in E}\sum_{j\in W} c_e(zf_{je} + zb_{je}) \tag{1}$$

$$\text{s.t.} \sum_{j\in W}(x_{ij} + y_{ij}) = 1 \qquad \forall i \in T \tag{2}$$

$$\sum_{e\in E_i}(p_{ije} + q_{ije}) = x_{ij} + y_{ij} \qquad \forall i \in T; \forall j \in W \tag{3}$$

$$\sum_{j\in W}(NW - j + 1)(x_{rj} - x_{sj}) \geq 0 \qquad \forall (r,s) \in PR \tag{4}$$

$$\sum_{j\in W}(NW - j + 1)(y_{sj} - y_{rj}) \geq 0 \qquad \forall (r,s) \in PR \tag{5}$$

$$\sum_{i\in T}\sum_{e\in E_i}(t_{ie0}p_{ije} + t_{ie1}q_{ije}) \leq CT * u_j \qquad \forall j \in W \tag{6}$$

$$p_{ije} + q_{ije} + x_{ij} - 1 \leq zf_{je} \qquad \forall i \in T; \forall j \in W; \forall e \in E_i \tag{7}$$

$$p_{ije} + q_{ije} + y_{ij} - 1 \leq zb_{je} \qquad \forall i \in T; \forall j \in W; \forall e \in E_i \tag{8}$$

$$\sum_{j\in W}(zf_{je} + zb_{je}) \leq NE_e \qquad \forall e \in E \tag{9}$$

$$\sum_{i\in T}\sum_{e\in E_i} q_{ije} \leq M * k_j \qquad \forall j \in W \tag{10}$$

$$\sum_{j\in W} k_j \leq NA \tag{11}$$

$$p_{ije}, q_{ije}, x_{ij}, y_{ij}, zf_{je}, zb_{je}, u_j, k_j \in \{0,1\} \qquad \forall i \in T; \forall j \in W; \forall e \in E_i \tag{12}$$

The objective function (1) is the total annual cost of workstation utilization, equipment operation and assistant employment on the U-shaped assembly line. Constraint set (2) restricts the assignment of each task to just one side (front or back) of one of the workstations. Constraint set (3) requires that each task be completely processed either without equipment or using only 1 equipment type. Further, a task should be processed either with or without an assistant. Constraint sets (4) and (5) are precedence constraints among tasks on the front side and back side, respectively, of the the assembly line. Constraint set (6) is the cycle time constraint for each workstation. Constraint sets (7) and (8) ensure that a task is processed on an equipment type at a workstation only if that equipment is assigned to that workstation. Constraint set (9) are equipment availability constraints. Constraint set (10) ensures that an assistant cannot be used to process a task at a workstation unless one is assigned to that workstation. Constraint (11) is the assistant availability constraint.

Constraint set (12) are binary constraints on the decision variables.

The mathematical model (1)-(12) of RDULB, for a problem of practical size, involves too many integer variables and constraints to be solved optimally even by the best commercial solver available in the industry. The IP formulation of a simple (i.e., not resource dependent) ULB proposed by Urban (1998) is reported to be too large to solve problem instances of the size beyond 45 tasks and 14 workstations. The IP formulation of RDULB, which involves far too many variables and constraints than a similar formulation for ULB, is even more difficult to solve optimally. Kara *et al.* [14] report difficulty solving the IP formulation for RDULB even for some problem instances of the size of 30 tasks and 9 equipment types.

The size of the model can be reduced to some extent by eliminating some of the variables and constraints through the use of appropriate bounds. For a straight assembly line balancing problem, lower and upper bounds on the workstation index to which a given task can be assigned may be used to eliminate variables that represent infeasible assignments. However, for a U-shaped assembly line, there is no upper bound for any task, since it is theoretically possible for all the assignments to be made through the phantom diagram (Urban, 1998). Nonetheless, we can still eliminate some of the variables through the use of lower bounds, computed as follows:

$$LB_i = \min\left\{\left\lceil\left(t_i + \sum_{k\in\{r:(r,i)\in PR\}} t_k\right)/CT\right\rceil, \left\lceil\left(t_i + \sum_{k\in\{s:(i,s)\in PR\}} t_k\right)/CT\right\rceil\right\} \qquad (13)$$

where, $t_i = \min_{e\in E_i}\{t_{ie0}, t_{ie1}\}$. Using this lower bound, W in (2), (3), (7), and (8) can be replaced by $W_i = [LB_i, NW]$. Similarly, W in (4) and (5) can be replaced by $[LB, NW]$, where:

$$LB = \min\{LB_r, LB_s\} \qquad (14)$$

The model can be further tightened by avoiding the use of Big M in the model. This can be done by finding an appropriate value that can be used for Big M in (10). We argue that NT represents a good value of Big M in (10) as follows. (2) and (3) together imply:

$$\sum_{e\in E_i}(p_{ije} + q_{ije}) \leq 1 \qquad \forall i \in T; \forall j \in W$$

Summing the above set of equations over T gives the following:

$$\sum_{i \in T} \sum_{e \in E_i} (p_{ije} + q_{ije}) \leq NT \qquad \forall j \in W$$

The above set of equations further implies:

$$\sum_{i \in T} \sum_{e \in E_i} q_{ije} \leq NT \qquad \forall j \in W$$

This suggests that Big M in (10) can be replaced by NT. Further, the following constraint may be added to ensure that workstation j is not used unless workstation $j-1$ is used. This is to avoid awkward symmetries in the problem, which unnecessarily increases the computation time.

$$u_j \leq u_{j-1} \forall j \in W \tag{15}$$

With the above substitutions, a tighter mathematical model for RDULB can be stated as: $[RDULB']$:

(1)

$$\text{s.t.} \sum_{j \in [LB_i, NW]} (x_{ij} + y_{ij}) = 1 \qquad \forall i \in T \qquad (2')$$

$$\sum_{e \in E_i} (p_{ije} + q_{ije}) = x_{ij} + y_{ij} \qquad \forall i \in T; \forall j \in [LB_i, NW] \qquad (3')$$

$$\sum_{j \in [LB, NW]} (NW - j + 1)(x_{rj} - x_{sj}) \geq 0 \qquad \forall (r,s) \in PR \qquad (4')$$

$$\sum_{j \in [LB, NW]} (NW - j + 1)(y_{sj} - y_{rj}) \geq 0 \qquad \forall (r,s) \in PR \qquad (5')$$

$$p_{ije} + q_{ije} + x_{ij} - 1 \leq z f_{je} \qquad \forall i \in T; \forall j \in [LB_i, NW]; \forall e \in E_i \qquad (7')$$

$$p_{ije} + q_{ije} + y_{ij} - 1 \leq z b_{je} \qquad \forall i \in T; \forall j \in [LB_i, NW]; \forall e \in E_i \qquad (8')$$

$$\sum_{i \in T} \sum_{e \in E_i} q_{ije} \leq NT * k_j \qquad \forall j \in W \qquad (10')$$

$(6), (9), (11) - (12), (15)$

where LB_i and LB are given by (13) and (14), respectively.

Our limited computational study does indicate improvement in the computation time of $[RDULB']$ compared to $[RDULB]$. However, $[RDULB']$ is still too difficult to solve for

some problem instances even of the size of 30 tasks and 9 equipment types. In the following section, we, therefore, develop a Simulated Annealing (SA) based solution that can solve large problem instances of $[RDULB]$.

3. Solution Algorithm

Our motivation to use SA to solve $[RDULB]$ comes from the literature on its successful application to a variety of difficult combinatorial optimization problems like the quadratic assignment problem [6], graph colouring problems [4], packing problems [8], scheduling problems [28] and cell formation problems [13]. Any SA algorithm has the following three important components [15]: (i) initial solution generation; (ii) neighborhood solution generation; and (iii) termination. The scheme used for each of these components with respect to $[RDULB]$ is described next.

3.1. Initial Solution Generation

Initial solution is generated by assigning tasks one by one, starting with any one of the tasks that do not have a predecessor, to only one side (front) of the assembly line, subject to the precedence and Cycle time constraints. A resource (an equipment or an assistant) is used sparingly, assigned to a workstation only when it is absolutely necessary to process a task assigned to that workstation. The minimum possible use of resources in the initial solution allows flexibility in constructing feasible neighborhood solutions in SA. Therefore, a tie between two or more tasks that are eligible (with respect to precedence and Cycle time constraints) for assignment to a given workstation is always broken in favour of the one that does not need an additional resource (an equipment or an assistant). Ties between two tasks, both of which do not require additional resources is broken arbitrarily. Similarly, ties between two tasks, both of which require additional resources is also broken arbitrarily. The above procedure for generating initial solution will thus satisfy all the constraints (2) - (12) and (15), but may require more than NW workstations since additional resources, which help save task processing times, are used as sparingly as possible. We, therefore, allow more than NW workstations to be used, if required by the above procedure for a feasible initial solution. This, of course, will make the solution suboptimal due to excess cost of too many workstations. The various steps of the initial solution generation scheme are illustrated using an example in section 4.

3.2. Neighborhood Solution Generation

A neighborhood solution is generated by randomly selecting any one of the following approaches:

- *Insert:* Randomly pick a task from a workstation and assign it to a randomly picked different workstation. Details are provided in Figure 4.
- *Swap:* Exchange two tasks randomly picked from two different workstations. Details are provided in Figure 5.
- Insert followed by swap.
- Swap followed by insert.

3.3. Termination

The algorithm terminates by any one of the following criteria:

- Temperature drops below a set threshold, T_{min}.
- Acceptance ratio (number of neighborhood solutions accepted/number of neighborhood solutions attempted) at any temperature reaches a minimum threshold value, R_f.
- No improvement in the objective function value for the last preset number of temperature transitions, $MAXITER_{\text{wi}}$.

The complete SA algorithm for RDULB is described in Figure 3. The details of Insert and Swap operations for neighborhood solution generation are shown in Figure 4 and Figure 5, respectively. In the following section, we present an illustrative example to demonstrate the various elements of SA algorithm, as applied to RDULB.

4. Illustrative Example

We explain the details of the proposed SA algorithm, presented in the previous section, using a small illustrative example involving 10 tasks (NT = 10) and 3 equipment types (NE = 3). The data on the precedence relations, possible alternative resources (equipment type and assistant) and the corresponding operation times for different tasks are presented in Table 1. Other equipment and cost related data are: $NE_e = \{1, 1, 2\}$ and $c_e = \{24, 16, 52\}$ for $e = \{1, 2, 3\}$; $cw = 100$, $ca = 70$, $CT = 45$ and $NW = 5$.

0. Read and initialize.

   0.1. Read RDULB parameters: NT, NW, NE, NE_e, NA, CT, E_i, PR, t_{ie0}, t_{ie1}, cw, ca, c_e, as defined in Section 2.

   0.2. Define Simulated Annealing parameters: initial temperature, T_0; maximum number of neighborhood solutions (transitions) attempted at each temperature, L_{max}; maximum number of neighborhood solutions (transitions) accepted at each temperature, AT_{max}; cooling factor, $0 < \alpha < 1$; minimum temperature, T_{min}; final acceptance ratio, R_f; $MAXITER_{\text{wi}}$ and $MAXTRIAL$.

   0.3. Initialize iteration (outer loop) counter, $i \leftarrow 0$; temperature, $T_i \leftarrow T_0$.

   0.4. Initialize inner loop counter, $l \leftarrow 0$; and accepted number of transitions, $AT \leftarrow 0$.

   0.5. Generate initial solution, SOL_l^i, as described in Section 3.1, and determine its objective function value, OBJ_l^i. $SOL^{best} \leftarrow SOL_l^i$ and $OBJ^{best} \leftarrow OBJ_l^i$.

1. Execute outer loop (1.1–1.4).

   1.1. Execute inner loop (1.1.1–1.1.5).

      1.1.1. $l \leftarrow l + 1$.

      1.1.2. Generate a neighborhood solution, SOL_l^i, by randomly choosing any one of the four options described in Section 3.2, and calculate its objective function value, OBJ_l^i. Repeat this until a feasible neighborhood solution is found or $MAXTRIAL$ attempts are reached. If no feasible solution is found in $MAXTRIAL$ attempts, then go to step 1.1.1.

      1.1.3. $\delta \leftarrow OBJ_l^i - OBJ_{l-1}^i$.

      1.1.4. If $\delta \leq 0$ or $r \sim U(0,1) \leq e^{(-\delta/T_i)}$, then go to step 1.1.4.1. Else, reject the neighborhood solution, $SOL_l^i \leftarrow SOL_{l-1}^i$ and $OBJ_l^i \leftarrow OBJ_{l-1}^i$, and go to step 1.1.5.

         1.1.4.1. Accept SOL_l^i and OBJ_l^i.

         1.1.4.2. $AT \leftarrow AT + 1$.

         1.1.4.3. If $OBJ_l^i < OBJ^{best}$, then $SOL^{best} \leftarrow SOL_l^i$ and $OBJ^{best} \leftarrow OBJ_l^i$.

      1.1.5. If one of the following conditions holds true: (a) $AT \geq AT_{max}$; (b) $l \leftarrow L_{max}$, then terminate the inner loop, $SOL_0^i \leftarrow SOL_l^i$ and $OBJ_0^i \leftarrow OBJ_l^i$, then $l \leftarrow 0$ and $AT \leftarrow 0$, and go to step 1.2. Otherwise, continue the inner loop and go to 1.1.1.

   1.2. $i \leftarrow i + 1$.

   1.3. Reduce the temperature: $T_i \leftarrow \alpha T_{i-1}$.

   1.4. If one of the following conditions holds true: (a) $T_i < T_{min}$; (b) $AT/l \leq R_f$; (c) the objective function value has not improved for the last $MAXITER_{\text{wi}}$ iterations, then terminate the outer loop and go to step 2. Else, continue the outer loop and go to step 1.1.

2. Return SOL^{best} with objective function value OBJ^{best}, and terminate the procedure.

Figure 3: Simulated Annealing (SA) Algorithm for RDULB

1. Randomly pick a task, say t. Let it belong to workstation, say $j1$.
2. Randomly pick a workstation, say $j2$ such that $j2 \neq j1$.
3. If feasible (with respect to precedence and cycle time constraints), release the resource(s) used by t by reassigning other tasks at $j1$ that share resources with t to other resources available at $j1$. The released resource(s) may be assigned to $j2$ in step 4 if that makes assignment of t to $j2$ feasible (with respect to precedence and cycle time constraints).
4. If feasible (with respect to precedence and cycle time constraints), move t from $j1$ to $j2$. Randomly decide whether to move t to front or back side of the workstation $j2$.
5. Allocate new resources to t at $j2$, if necessary and available.

Figure 4: Scheme for Insert Operation

1. Randomly pick 2 tasks, say $t1$ and $t2$ such that $t2 \neq t1$ from workstations, say $j1$ and $j2$, respectively such that $j2 \neq j1$.
2. If feasible (with respect to precedence and cycle time constraints), release the resource(s) used by $t1$ by reassigning other tasks at $j1$ that share resources with $t1$ to other resources available at $j1$. The released resource(s) may be assigned to $j2$ in step 3 if that makes assignment of $t1$ to $j2$ feasible (with respect to precedence and cycle time constraints). Similarly, if feasible, release the resource(s) used by $t2$. The released resource(s) may be assigned to $j1$ in step 3 if that makes assignment of $t2$ to $j1$ feasible.
3. If feasible (with respect to precedence and cycle time constraints), move $t1$ from $j1$ to $j2$, and $t2$ from $j2$ to $j1$. For this, randomly choose one of the following options.
   (a) assign $t1$ to front of $j2$ and $t2$ to front of $j1$.
   (b) assign $t1$ to back of $j2$ and $t2$ to back of $j1$.
   (c) assign $t1$ to front of $j2$ and $t2$ to back of $j1$.
   (d) assign $t1$ to back of $j2$ and $t2$ to front of $j1$.
4. Allocate new resources to $t1$ at $j2$, if necessary and available. Similarly, allocate new resources to $t2$ at $j1$, if necessary and available.

Figure 5: Scheme for Swap Operation

The various steps in the generation of initial solution are detailed in Table 2. To begin with, tasks $1, 2, 3$ and 7 are all eligible for assignment to workstation 1 since none of them has any precedence constraint, and any of them can fit in the workstation in the given cycle time. Of these, tasks $1, 3$ and 7 are preferred since they can be processed without any equipment or assistant. As described in section 3.1, this is done to minimize the possible use of resources in the initial solution, which allows flexibility in constructing feasible neighborhood solutions. Of the tasks $1, 3$ and 7, task 1 is selected randomly. Assigning task 1 to workstation 1 makes task 4 also eligible for assignment. Of tasks $2, 3, 4$ and 7 that are now eligible, tasks $3, 4$ and 7 are preferred since they can be processed without any equipment or assistant. Of them, task 4 is randomly selected in step 2. Tasks in the remaining steps are similarly selected. In step 5, tasks 2 and 7 satisfy the precedence constraint, but only 7 can fit in the remaining time available at workstation 1. However, task 7 is not assigned to workstation 1 since it will necessarily need an assistant to be completed within the remaining time available at workstation 1. Hence, task 7 is assigned to workstation 2, which does not necessarily require an assistant to be completed within the cycle time.

The initial solution obtained is presented in Figure 6. The total cost with the initial solution is 370. The final solution of the SA algorithm is presented in Figure 7. The final SA solution employs 2 workstations, 1 assistant and 1 unit of equipment type 1, giving a total cost of 294, which is optimal (confirmed by solving the resulting mathematical model using Cplex).

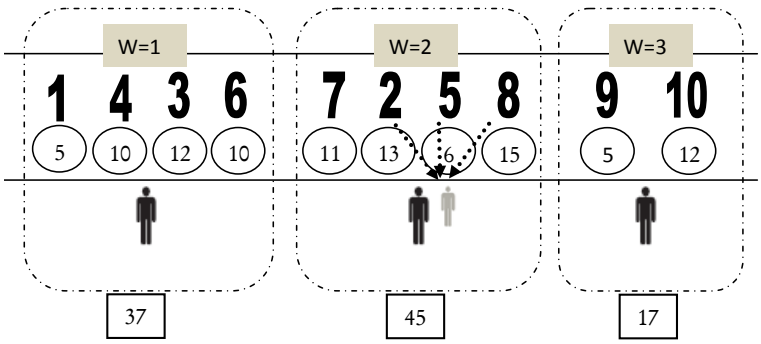

Figure 6: Initial Solution for Illustrative Example

5. Computational Experience

We test the efficiency of the proposed SA algorithm and the quality of solution produced by it by attempting to solve a variety of problem instances of RDULB using SA as well as

Table 1: Task Time Data for Illustrative Example

Task	Immediate Predecessors	Assistant Assigned?	Equipment Type 0	Equipment Type 1	Equipment Type 2	Equipment Type 3
1	-	No	5			
		Yes				
2	-	No				
		Yes	13			
3	-	No	12	8		
		Yes				
4	1	No	10			
		Yes				
5	2	No	9			
		Yes	6			
6	3	No	10			8
		Yes				
7	-	No	11			
		Yes	8			
8	4, 5	No				
		Yes	15	8		10
9	6, 8	No	5		3	4
		Yes				
10	7, 9	No	12			
		Yes				

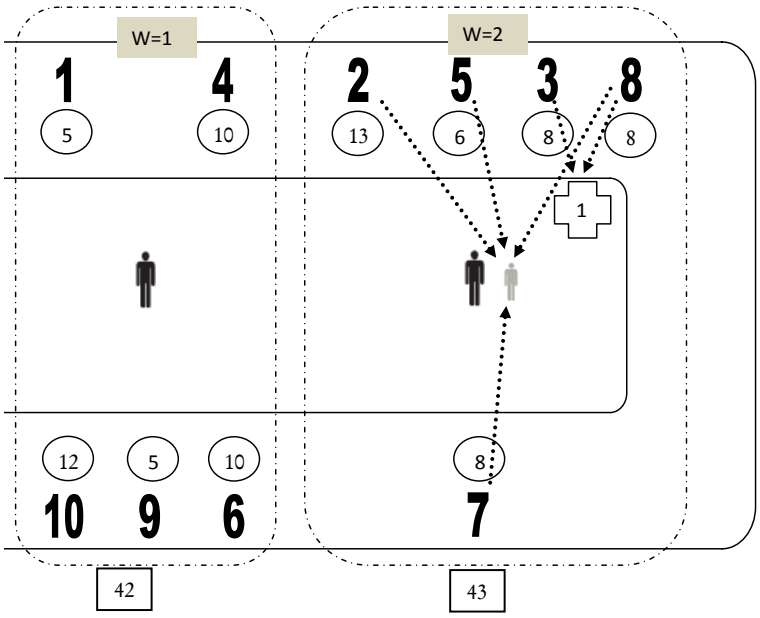

Figure 7: Final SA Solution for Illustrative Example

using Cplex for the IP model. The SA algorithm is coded and compiled in Visual C++ while the IP model is solved using Ilog Cplex 12.1 Concert Technology in Visual C++. All the computational experiments are done on a personal computer with 2 GB RAM and 2.4 GHz, Core 2 Duo Intel processor. The parameters of SA algorithm play a crucial

Table 2: Initial Solution for Illustrative Example

Work station	**Time Available**	**Tasks satisfying precedence constraints**	**Task(equipment) [Assistant] combinations that can fit**	**Task Assigned**	**Equipment Assigned**	**Assistant Assigned?**	**Time Left**
1	45	1, 2, 3, 7	1(0)[No]; 2(0)[Yes]; 3(0)[No]; 3(1)[No]; 7(0)[No]; 7(0)[Yes]	1	0	No	40
1	40	2, 3, 4, 7	2(0)[Yes]; 3(0)[No]; 3(1)[No]; 4(0)[No]; 7(0)[No]; 7(0)[Yes]	4	0	No	30
1	30	2, 3, 7	2(0)[Yes]; 3(0)[No]; 3(1)[No]; 7(0)[No]; 7(0)[Yes]	3	0	No	18
1	18	2, 6, 7	2(0)[Yes]; 6(0)[No]; 6(3)[No]; 7(0)[No]; 7(0)[Yes]	6	0	No	8
1	8	2, 7	7(0)[Yes]	None	-	-	8
2	45	2, 7	2(0)[Yes]; 7(0)[No]; 7(0)[Yes]	7	0	No	34
2	34	2	2(0)[Yes]	2	0	Yes	21
2	21	5	5(0)[No]; 5(0)[Yes]	5	0	Yes	15
2	15	8	8(0)[Yes]; 8(1)[Yes]; 8(3)[Yes]	8	0	Yes	0
3	45	9	9(0)[No]; 9(2)[No]; 9(3)[No]	9	0	No	40
3	40	10	10(0)[No]	10	0	No	28

role in its performance. There is a trade-off between the solution quality and solution time. SA parameters are finalized after testing with different combinations of values. The final SA parameter values chosen for different number of tasks are shown in Table 3. The remaining parameters are set as: $AT_{max} = L_{max}/2$; $T_0 = X \times (cw + ca + \max_e\{c_e\})$; $T_{min} = (\min\{cw, ca, c_e\})/4$; $MAXTRIAL = NT$; $MAXITER_{\text{wi}} = L_{max}$.

Table 3: SA Parameters

NT	L_{max}	α	R_f	X
10	200	0.6	0.02	2
20	600	0.85	0.02	100
30	800	0.95	0.02	500
40	1000	0.96	0.02	1000
50	1200	0.97	0.02	4000
≥ 89	1200	0.98	0.02	8000

We divide the problem instances into three categories: (i) small-to-medium instances, consisting of 10, 20 and 30 tasks, and 3, 6 and 9 equipment types, respectively; (ii) large instances, consisting of 40 and 50 tasks, and 12 and 15 equipment types, respectively; and (iii) very large instances, consisting of 89, 94, 111 and 148 tasks, and 15 or 18 equipment types. The small-to-medium and large problem instances are used directly from Kara *et al.* [14]. For a specified number of tasks and equipment types, the data sets for small-to-medium and large problem instances are available for 3 different precedence diagrams. Each precedence diagram corresponds to a given value of flexibility ratio (FR) ((i) FR = 0.25 ± 0.05; (ii) FR = 0.50 ± 0.05; (iii) FR = 0.75 ± 0.05) - an indicator of complexity of precedence relationships ([24]. For each data set, the problem is solved for 3 different cycle times (CT = 30, 45, 60), each for 3 different assignments of task times. This gives us 81 ($3 \times 3 \times 3 \times 3$) small-to-medium and 54 ($2 \times 3 \times 3 \times 3$) large problem instances. The scheme used to construct the task time and cost data for these problem instances are as given below.

- $NE_e \sim U(0, 2)\ \forall e \in \{E : e \neq 0\}$.
- $t_{ie0} \sim U(1, 15)$ and $t_{ie1} \sim U(1, 15)\ \forall e \in E_i$.
- $t_{ie1} < t_{ie0}\ \forall e \in E_i$; $t_{ie0} < t_{i00}$ and $t_{ie1} < t_{i01}\ \ \forall e \in \{E_i : e \neq 0\}$.
- $cw = 100$ and $ca = 70$.
- $c_e \sim U(10, 60)\ \forall e \in \{E : e \neq 0\}$.

The computational results for small-to-medium problem instances are presented in Table 4. As obvious from Table 4, problem instances even of the size of 30 tasks may become too difficult for Cplex to solve: it is unable to solve 2 such instances. Of the remaining 79 instances, the proposed SA algorithm is able to solve 55 of them to optimality within the CPU time taken by Cplex. Of the remaining 24, the optimality gap is within 5% for 14 of them, and within 10% for all of them.

Cplex is, however, unable to find an optimal solution to most of the larger problem instances (40 and 50 tasks) within a time limit of 3 hours. So, we solve these instances using the proposed SA algorithm, and then look for the best feasible solution produced by Cplex at times T, 2T, 4T, 6T, 8T and 10T, where T is the CPU time taken by SA algorithm. The results for these larger problem instances are presented in Table 5, from which the following observations can be made: (i) Of the 54 large problem instances, Cplex could solve to optimality (a) only 10 instances within the same CPU time as SA; (b) only 11 instances within 4 times the CPU time taken by SA; (c) only 15 instances within 10 times the same CPU time taken by SA. Cplex could not find an optimal solution in the remaining 39 instances even after 10 times the CPU time taken by SA; (ii) Of these 39 instances, Cplex produced worse or at best as good solution as SA (indicated by a non-negative % gap in Table 5) for (a) 22 instances within the same CPU time as SA; (b) 15 instances within 2 times the CPU time taken by SA; (c) 11 instances within 4 times the CPU time taken by SA; (d) 10 instances within 6 times the CPU time taken by SA; (e) 7 instances within 8 times the CPU time taken by SA; (f) 7 instances within 10 times the CPU time taken by SA.

The very large problem instances are adapted from the Simple Assembly Line Balancing Problem (SALBP) instances for different number of tasks (89, 94, 111 and 148) available at www.assembly-line-balancing.de – a general platform for research on assembly line design and scheduling. For such instances, the additional data for alternative resource requirements are generated randomly using the following scheme.

- $NE_e \sim U(0,2)\ \forall e \in \{E : e \neq 0\}$.
- $t_{ie0} = (100 - d\%)t_{i00}\ \ \forall e \in \{E_i : e \neq 0\}$ and $t_{ie1} = (100 - d\%)t_{ie0}\ \ \forall e \in E$, where $d \sim U(20,50)$.
- $cw = 100$ and $ca = 70$.
- $c_e \sim U(10,60)\ \forall e \in \{E : e \neq 0\}$.

Each instance corresponding to a given number of tasks is solved for 3 different cycle times, giving a total of 12 different instances. The computational results for these very large problem instances are presented in Table 6. As obvious from Table 6, Cplex is unable to find a single feasible solution for 11 of the 12 problem instances even after 10 times the CPU time taken by SA. In contrast, the proposed SA algorithm produces a good feasible solution (with around 19% improvement, on average, from the randomly generated initial solution) in between half an hour to 1.5 hours.

6. Conclusions & Future Research

In this paper, we developed a Simulated Annealing (SA) based algorithm to solve large instances of the resource dependent U-shaped assembly line balancing (RDULB) problem, which are otherwise too difficult to solve for the commercial mathematical programming solvers even for problems of the size of 30 tasks and 9 equipment types. The proposed SA algorithm is able to solve most of the small-to-moderate size problem instances to optimality or close to optimality very efficiently. A comparison with Cplex for a variety of larger problem instances suggests that Cplex is unable to beat the proposed SA even after 10 times the CPU time taken by SA. For very large problem instances (89 tasks and above), SA produces a good feasible solution (with 19% improvement, on average, from a randomly generated initial solution) within half an hour to 1.5 hours, while Cplex fails to produce even a single feasible solution even after 10 times the CPU time taken by SA.

We have demonstrated in this paper that the proposed SA clearly outperforms traditional Branch & Bound or Branch & Cut methods used by Cplex for large problem instances of RDULB. Nonetheless, like all other heuristic algorithms, the proposed SA algorithm also suffers from the drawback that it does not give any information on the optimality gap of its solutions unless the problem can be solved optimally by Cplex. Thus, developing a good lower bound for RDULB is an obvious (but challenging) avenue for future research. Our effort in this direction has so far given us little success. Use of other metaheuristic approaches like Genetic Algorithm or Tabu Search is another possible avenue for future research.

Acknowledgements

We would like to sincerely thank Dr. Yakup Kara from Selcuk University, Turkey for kindly sharing some of the problem data used in this paper.

Table 4: Computational Results for Small-to-Medium Problem Instances

NT	FR	NA	NE	CT	Kmax	Cplex Solution Optimal Solution (OS)	Cplex Solution Time (sec)	SA Solution SA Solution (SAS)	SA Solution Time (sec)	% Optimality Gap = 100*(OS - SAS)/OS
10	0.25	2	3	30	5	370	1.006	370	1.138	0.0000
					5	370	4.33	370	0.904	0.0000
					5	370	2.218	370	0.764	0.0000
	0.50	2	3	30	5	354	1.61	354	1.731	0.0000
					5	410	1.197	410	2.433	0.0000
					5	410	0.881	410	0.983	0.0000
	0.75	2	3	30	5	413	1.203	413	2.418	0.0000
					5	382	1.246	382	3.993	0.0000
					5	370	5.451	370	1.716	0.0000
20	0.25	4	6	30	6	754	27.058	756	68.124	0.2653
					6	584	65.701	584	21.512	0.0000
					6	754	15.488	779	22.37	3.3156
	0.50	4	6	30	6	711	20.718	737	75.784	3.6568
					6	508	95.803	524	47.502	3.1496
					6	586	111.055	598	53.118	2.0478
	0.75	4	6	30	6	751	10.955	751	30.388	0.0000
					6	533	8.446	533	59.276	0.0000
					6	681	2.885	681	64.368	0.0000
30	0.25	6	9	30	9	940	312.484	940	164.268	0.0000
					10	-	>3hrs	1088	239.304	-
					9	789	214.027	863	715.309	9.3790
	0.50	6	9	30	9	831	233.592	864	437.257	3.9711
					11	1041	1856.24	1062	673.47	2.0173
					10	895	1313.98	961	374.75	7.3743
	0.75	6	9	30	10	893	2493.67	917	303.89	2.6876
					10	988	1969.03	1040	135.21	5.2632
					11	-	>3hrs	1071	121.918	-
10	0.25	2	3	45	5	270	0.448	270	1.092	0.0000
					5	270	0.584	270	0.604	0.0000
					5	270	0.809	270	0.619	0.0000
	0.50	2	3	45	5	270	1.568	270	0.538	0.0000
					5	294	1.933	294	0.498	0.0000
					5	310	0.972	310	0.648	0.0000
	0.75	2	3	45	5	282	0.471	282	0.605	0.0000
					5	282	1.766	282	0.637	0.0000
					5	270	0.325	270	0.64	0.0000
20	0.25	4	6	45	5	540	76.83	540	4.094	0.0000
					5	430	25.627	470	5.002	9.3023
					5	540	48.018	556	5.673	2.9630
	0.50	4	6	45	5	495	104.692	499	6.698	0.8081
					5	370	58.112	370	6.113	0.0000
					5	399	13.526	399	7.646	0.0000
	0.75	4	6	45	5	481	1.272	481	8.253	0.0000
					5	370	41.313	370	6.411	0.0000
					5	470	21.721	470	13.416	0.0000
30	0.25	6	9	45	6	640	617.464	640	99.317	0.0000
					7	730	94.442	776	133.216	6.3014
					6	570	171.569	570	142.119	0.0000
	0.50	6	9	45	6	633	621.586	633	120.468	0.0000
					7	721	75.566	740	128.518	2.6352
					6	603	294.122	615	127.799	1.9900
	0.75	6	9	45	6	570	66.128	570	95.416	0.0000
					6	740	301.532	740	84.246	0.0000
					6	740	322.842	740	84.623	0.0000
10	0.25	2	3	60	2	270	0.128	270	1.716	0.0000
					2	270	0.191	270	0.592	0.0000
					2	235	0.343	235	2.043	0.0000
	0.50	2	3	60	2	238	0.417	238	2.09	0.0000
					2	270	0.285	270	0.67	0.0000
					2	310	0.403	310	0.624	0.0000
	0.75	2	3	60	2	282	0.355	282	0.686	0.0000
					2	270	0.448	270	0.655	0.0000
					2	270	0.359	270	0.717	0.0000
20	0.25	4	6	60	4	440	7.049	440	16.848	0.0000
					4	370	1.921	370	14.398	0.0000
					4	440	1.435	440	16.083	0.0000
	0.50	4	6	60	4	395	9.158	399	17.16	1.0127
					4	295	7.138	296	31.371	0.3390
					4	311	0.778	311	173.759	0.0000
	0.75	4	6	60	4	381	0.884	381	15.147	0.0000
					4	312	1.612	312	97.141	0.0000
					4	370	6.007	370	41.184	0.0000
30	0.25	6	9	60	5	540	81.244	540	27.939	0.0000
					5	559	9.126	606	293.104	8.4079
					5	470	2.636	470	39.592	0.0000
	0.50	6	9	60	5	470	1.622	508	31.059	8.0851
					5	588	14.519	592	51.448	0.6803
					5	470	2.197	503	47.408	7.0213
	0.75	6	9	60	5	470	1.06	470	20.748	0.0000
					5	534	19.671	570	41.558	6.7416
					5	540	28.314	587	258.565	8.7037

- indicates Cplex is unable to find the optimal solution within a 3 hours limit.

Table 5: Computational Results for Large Problem Instances

						SA Solution		Cplex Solution (CS) % Gap = 100*(CS - FSAS)/FSAS					
NT	**FR**	**NA**	**NE**	**CT**	**Kmax**	**Final SA Solution (FSAS) SA Solution**	**Time, T (sec)**	**At T (sec)**	**At 2T (sec)**	**At 4T (sec)**	**At 6T (sec)**	**At 8T (sec)**	**At 10T (sec)**
40	0.25	8	12	30	13	1310	1003.54	5.34	0.00	0.00	0.00	0.00* (6960.31)	
					12	1183	564.77	-5.75	-7.27	-7.27	-7.27	-7.27	-7.27
					14	1461	802.84	3.01	-1.78	-1.78	-1.78	-2.26	-2.26
	0.50	8	12	30	14	1310	518.25	2.14	0.00	0.00	0.00	0.00	0.00
					12	1160	548.38	-5.34	-5.34	-5.34	-5.34	-5.34	-5.34
					13	1255	556.79	0.00	-1.20	-1.20	-1.20	-1.20	-1.20
	0.75	8	12	30	14	1362	577.72	-1.40	-2.79	-2.79	-2.79	-2.79	-2.79
					12	1210	517.16	0.00	0.00	-0.17	-0.17	-0.17	-0.17
					15	1423	411.13	0.98	0.98	0.00	-0.35	-1.48	-1.55
50	0.25	10	15	30	15	1332	1524.83	3.45	3.45	0.08	0.08	0.08	0.08
					18	1767	4279.47	0.74	-3.23	-3.23	-3.23	-3.23	-3.23
					17	1656	1970.00	1.99	-0.66	-3.44	-3.44	-3.44	-3.44
	0.50	10	15	30	18	1870	1730.43	2.25	0.53	0.53	0.53	-0.37	-0.37
					18	1812	1577.93	4.14	2.48	-3.53	-3.53	-3.53	-3.53
					17	1671	1154.63	0.72	-3.65	-3.65	-3.65	-3.65	-3.65
	0.75	10	15	30	16	1580	1184.43	5.13	0.70	0.70	0.70	0.70	0.70
					17	1632	1344.44	0.00	-0.67	-0.67	-0.67	-0.67	-0.67
					17	1680	1348.15	4.17	-4.17	-4.17	-4.17	-4.17	-4.17
40	0.25	8	12	45	8	928	381.47	-1.94	-1.94	-1.94	-1.94	-1.94	-1.94
					8	770	293.23	0.00	0.00	0.00	0.00	0.00	0.00
					8	1010	403.69	-4.46	-6.93	-6.93	-6.93	-6.93	-6.93* (3735.01)
	0.50	8	12	45	8	940	255.31	-9.15	-9.15	-9.15	-9.15	-9.15	-9.15
					8	770	237.86	0.00	0.00	0.00	0.00	0.00	0.00
					8	917	250.82	-6.76	-6.76	-6.76	-6.76	-6.87	-6.87* (2168.73)
	0.75	8	12	45	8	940	341.16	-5.74	-5.74	-5.74	-5.74	-5.74	-5.74
					8	840	235.53	0.00	0.00	0.00	0.00	0.00	0.00
					8	953	392.24	0.00	-1.36	-1.36	-1.36	-1.36	-1.36* (3760.35)
50	0.25	10	15	45	10	1010	777.38	-5.45	-5.45	-7.62	-7.62	-7.62	-7.62* (6670.39)
					10	1210	938.76	-5.79	-5.79	-5.79	-5.79	-5.79	-5.79
					10	1140	824.40	0.00	0.00	0.00	0.00	-1.84	-4.82
	0.50	10	15	45	10	1263	770.57	1.35	1.35	-4.20	-4.20	-4.20	-4.20
					10	1210	810.39	0.00	0.00	0.00	0.00	0.00	0.00
					10	1140	833.69	-2.63	-2.63	-2.63	-2.63	-2.63	-2.63
	0.75	10	15	45	10	1110	658.77	-7.03	-8.02	-8.02* (2463.61)			
					10	1210	702.45	-6.45	-7.36	-7.36	-7.36	-7.36	-7.36
					10	1210	723.39	-5.79	-5.79	-5.79	-5.79	-5.79	-5.79
40	0.25	8	12	60	6	740	435.88	-4.86* (183.456)					
					5	588	334.53	-3.06* (16.347)					
					6	740	356.24	0.00* (21.301)					
	0.50	8	12	60	6	740	489.95	-7.84* (708.273)					
					5	570	327.39	0.00* (16.692)					
					6	740	247.62	-7.57* (78.511)					
	0.75	8	12	60	6	740	403.58	-0.14* (100.52)					
					6	640	385.90	-5.94* (54.652)					
					6	740	506.47	0.00* (74.661)					
50	0.25	10	15	60	7	810	883.75	-8.64	-8.64	-8.64	-8.64	-8.64	-8.64
					8	940	1180.83	0.00	0.00	-0.11	-1.28	-2.13	-2.13
					8	863	1050.24	-2.67	-2.67	-2.67	-2.67	-2.67	-2.67
	0.50	10	15	60	8	1010	1029.10	-6.93	-6.93	-6.93	-6.93	-6.93	-6.93
					8	1052	810.15	-17.21	-17.21	-17.21	-17.21	-17.21	-17.21
					8	940	793.77	-10.64	-10.64	-10.64	-10.64	-10.64	-10.64
	0.75	10	15	60	7	840	909.07	-2.26	-3.57	-3.57	-3.57	-3.57	-3.57
					9	851	818.40	-1.29	-1.29	-1.29	-1.29	-1.29	-1.29
					8	840	2730.33	0.00 * (233.426)					

$*$ indicates Cplex solution is optimal; number within parenthesis indicates the corresponding CPU time.

Table 6: Computational Results for Very Large Problem Instances

						SA Solution				Cplex Solution (CS) % Gap = 100*(CS - FSAS)/FSAS					
Instance	**NT**	**NA**	**NE**	**CT**	**Kmax**	**Initial SA Solution (ISAS)**	**Final SA Solution (FSAS)**	**Time, T (sec)**	**% improvement = 100*(ISAS - FSAS)/ISAS**	**At T (sec)**	**At 2T (sec)**	**At 4T (sec)**	**At 6T (sec)**	**At 8T (sec)**	**At 10T (sec)**
Lutz3	89	10	15	75	25	3200	2542	3977.95	20.56	-	-	-	-	-	-
				100	19	2290	1924	3405.42	15.98	-	-	-	-	-	-
				150	15	1760	1518	2722.07	13.75	17.98	17.98	2.11	-4.41	-6.19	-20.29
Mukherjee	94	10	15	200	25	2890	2384	1940.98	17.51	-	-	-	-	-	-
				250	20	2150	1966	1580	8.56	-	-	-	-	-	-
				300	16	1990	1670	1386.26	16.08	-	-	-	-	-	-
ARC111	111	12	18	7000	30	3160	2484	2700	21.39	-	-	-	-	-	-
				8000	23	2930	2223	2599.47	24.13	-	-	-	-	-	-
				8500	23	2900	2078	2243.48	28.34	-	-	-	-	-	-
Barthol2	148	15	18	150	32	3980	3326	4912.27	16.43	-	-	-	-	-	-
				200	25	3310	2552	3706.21	22.90	-	-	-	-	-	-
				250	18	2710	2065	3654	23.80	-	-	-	-	-	-

- indicates Cplex is unable to find any feasible solution

References

[1] Aase G R, Olson J R and Schniederjans M J (2004). U-shaped assembly line layouts and their impact on labor productivity: An experimental study. *European Journal of Operational Research* **156**(3): 698–711.

[2] Becker C and Scholl A (2006). A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research* **168**(3): 694–715.

[3] Bukchin J and Tzur M (2000). Design of flexible assembly line to minimize equipment cost. *IIE Transactions* **32**(7): 585–598.

[4] Chams M, Hertz A and Werra D de (1987). Some experiments with simulated annealing for colouring graphs. *European Journal of Operational Research* **32**(2): 260–266.

[5] Cheng C, Miltenburg J and Motwani J (2000). The effect of straight- and U-shaped lines on quality. *IEEE Transactions on Engineering Management* **47**(3): 321–334.

[6] Connolly D T (1990). An improved annealing scheme for the QAP. *European Journal of Operational Research* **46**(1): 93–100.

[7] Corominas A, Pastor R and Plans J (2008). Balancing assembly line with skilled and unskilled workers. *Omega* **36**(6): 1126–1132.

[8] Dowsland K A (1993). Some experiments with simulated annealing techniques for packing problems. *European Journal of Operational Research* **68**(3): 389–399.

[9] Erel E, Sabuncuoglu I and Aksu B A (2001). Balancing of U-type assembly systems using simulated annealing. *International Journal of Production Research* **39**(13): 3003–3015.

[10] Faaland B H, Klastorin T D, Schmitt T G and Shtub A (1992). Assembly Line Balancing with Resource Dependent Task Times. *Decision Sciences* **23**(2): 343–364.

[11] Gokcen H and Agpak K (2006). A goal programming approach to simple U-line balancing problem. *European Journal of Operational Research* **171**(2): 577-585.

[12] Gokcen H, Agpak K, Gencer C and Kizilkaya E (2005). A shortest route formulation of simple U-type assembly line balancing problem. *Applied Mathematical Modelling* **29**(4): 373–380.

[13] Jayaswal S and Adil G K (2004). Efficient algorithm for cell formation with sequence data, machine replications and alternative process routings. *International Journal of Production Research* **42**(12): 2419–2433.

[14] Kara Y, Ozguven C, Yalcin N and Atasaguna Y (2011). Balancing straight and U-shaped assembly lines with resource dependent task times. *International Journal of Production Research* **49**(21): 6387–6405.

[15] Kirkpatrick S, Gelatt C D and Vecchi M P (1983). Optimization by simulated annealing. *Science* **220**(4598): 671–680.

[16] Miltenburg G J and Wijngaard J (1994). The U-line line balancing problem. *Management Science* **40**(10): 1378–1388.

[17] Miltenburg J and Sparling D H (1994). *Optimal solution algorithms for the U-line balancing problem*. Working Paper, McMaster University, Hamilton, Canada.

[18] Miltenburg J (2000). The effect of breakdowns on U-shaped production lines. *International Journal of Production Research* **38**(2): 353–364.

[19] Miltenburg J (2001). U-shaped production lines: A review of theory and practice. *International Journal of Production Economics* **70**(3): 201–214.

[20] Monden Y (1998). *Toyota production system - An integrated approach to just-in-time*. Kluwer: Dordrecht.

[21] Moon I, Logendran R and Lee J (2009). Integrated assembly line balancing with resource restrictions. *International Journal of Production Research* **47**(19): 5525–5541.

[22] Nicosia G, Pacciarelli D and Pacifici A (2002). Optimally balancing assembly lines with different workstations. *Discrete Applied Mathematics* **118**(1-2): 99–113

[23] Pinto P A, Dannenbring D G and Khumawala B M (1983). Assembly line balancing with processing alternatives: An application. *Management Science* **29**(7): 817–830.

[24] Scholl A (1999). *Balancing and sequencing assembly lines*. Physica: Heidelberg.

[25] Scholl A and Klein R (1999). ULINO: Optimally balancing U-shaped JIT assembly lines. *International Journal of Production Research* **37**(4): 721-736.

[26] Toklu B and Ozcan U (2008). A fuzzy goal programming model for the simple U-line balancing problem with multiple objectives. *Engineering Optimization* **40**(3): 191–204.

[27] Urban T L (1998). Note. Optimal balancing of U-shaped assembly lines. *Management Science* **44**(5): 738–741.

[28] Van Laarhoven P J M, Aarts E H L and Lenstra J K (1992). Job shop scheduling by simulated annealing. *Operations Research* **40**(1): 113–125.