International Journal of Industrial Engineering, 8(3), 230-240, 2001.

THE SIMPLE ASSEMBLY LINE BALANCING PROBLEM WITH PARALLEL WORKSTATIONS - A SIMULATED ANNEALING APPROACH

Ana S. Simaria and Pedro M. Vilarinho

Departamento de Economia, Gestão e Engenharia Industrial
Universidade de Aveiro
3810-193 Aveiro, Portugal
Email: Ana S. Simaria, simaria@egi.ua.pt

This work presents a new mathematical programming model for the simple assembly line balancing problem with parallel workstations that allows the user to control the way workstations are 'parallelised'. As in the conventional procedures the user can limit the number of replicas allowed for each workstation, but in this novel approach the user can also define a minimum task time to trigger the replication of the workstations. Another important characteristic of the model is that it simultaneously minimises the number of workstation and smoothes the workload among them. Due to the model complexity a simulated annealing approach is proposed to solve it. An example is used for illustration and computational experience is provided. The computational tests show that the performance of the proposed procedure is quite good, taking into account the problem complexity.

Significance: Assembly lines are employed for high volumes of output, so the global savings that result from a small reduction in unit costs, fostered by a correctly balanced line, can be important. The proposed model introduces a more realistic approach to assembly line balancing than previous ones.

Keywords: Assembly line balancing, Mathematical programming, Simulated annealing.

(Received 16 September 2000 ; Accepted in revised form 24 May 2001)

1. INTRODUCTION

A simple assembly line is a set of sequential workstations connected by a material handling system, designed to carry out the assembly of a homogeneous product. In each workstation a set of tasks are performed using a predefined assembly process. In this assembly process the following issues are defined: (i) the time required to perform each task - task time, (ii) a set of precedence relationships, which determines the sequence in which the tasks can be performed and (iii) a set of zoning constraints, which force or forbid the assignment of different tasks to the same workstation.

In a paced assembly line (the type of line addressed in this research) each workstation has a predefined amount of time to complete all the tasks assigned to it - cycle time. When this time is elapsed the sub-assembly must be moved to the next workstation along the line and the workstation receives a new sub-assembly from the previous workstation. So, the cycle time determines the production rate of the assembly line.

The major problem when designing (or redesigning) an assembly line is the determination of a feasible assignment of the tasks to the workstations, so that both the precedence and zoning constraints are met. This problem is termed the simple assembly line balancing problem (SALBP), and it can be classified into two different types, according to the goal pursued:

(i) SALBP1 - minimises the number of workstations, for a given cycle time.
(ii) SALBP2 - minimises the cycle time, for a given number of workstations.

In the SALBP1 the cycle time, and consequently the production rate, has to be pre–specified, so it is more frequently used in the design of a new assembly line for which the demand can be easily forecasted. The SALBP2 deals with maximising the production rate on an existing assembly line, and so it is particularly useful when, for example, modifications of the assembly process require changes in the line.

Salveson (1955) was the first author to address the assembly line balancing problem (of the SALBP1 type) and since then it has been a prolific topic of research. Scholl (1999) provides an extensive review of the subject.

One of the most common assumptions usually incorporated in the techniques to solve the assembly line balancing problem is that each task can only be performed in one workstation and, so, the production rate is limited by the longest task time. When this assumption is relaxed parallel workstations can be introduced, and two or more replicas of a workstation can perform the same set of tasks on different assemblies. The introduction of parallel workstations not only allows for cycle times shorter than the longest task time and thus an increase in the production rate, but also provides greater flexibility in designing the assembly line (Buxey, 1974).

One of the main advantages of using an assembly line is that it can rely on low skilled labour that can be easily trained, due to the strict division of labour. When parallel workstations are introduced the number of tasks performed by each worker increases - the limit being an assembly line with only one workstation and a sufficient number of replicas to match the demand, in which each worker would perform all the tasks in the assembly process. So, in order to maintain that advantage, it is necessary to control the 'parallelisation' process. Other constraints that may justify the control of this process relate to economic and physical space issues (e.g., when additional equipment is required).

Several models for the SALBP1 with parallel workstations have been proposed in the literature (e.g. Johnson (1983), Pinto et al. (1975, 1981), Bard (1989), Daganzo and Blumenfeld (1994)), however, no control of the 'parallelisation' process is included. The introduction of parallel workstations in the assembly line is usually based on the trade-off between the cost of procuring additional equipment to set-up parallel workstations and the cost of hiring additional labour for a non-parallel line, so that demand is met. In the literature paralleling is allowed even when all tasks have processing times shorter than the cycle time. Schofield (1979) and Sarker and Shantikumar (1983) approach the control of the 'parallelisation' process by defining a limit on the number of parallel workstations, while Buxey (1974) includes a limit on the number of tasks per workstation.

In this paper, a mathematical programming model for the SALBP1 with parallel workstations and zoning constraints is presented. The model allows the decision-maker to limit not only the maximum number of replicas of a workstation, but also the conditions under which a workstation can be replicated. Due to the combinatorial nature of the problem a simulated annealing algorithm was developed to solve it, which provides sub-optimal solutions with reasonable computational effort.

2. MATHEMATICAL PROGRAMMING MODEL

The proposed model for the assembly line balancing with parallel workstations is based on the following set of assumptions:

(i) an homogeneous product is assembled by performing N tasks of a predefined assembly process,
(ii) the demand, over the planning horizon, requires the assembly line to be operated with a cycle time C,
(iii) the tasks (i=1,…,N) are performed on a set of workstations (k=1,…,S),
(iv) the time required to perform each task, t_i, is deterministic,
(v) the set of tasks that cannot be performed before task i is completed, F_i, (successors of task i) is given by the precedence constraints for the assembly process,
(vi) a successor of task i cannot be assigned to a workstation until task i has been assigned,
(vii) a task can be assigned to only one workstation,
(viii) a workstation can be duplicated up to a maximum of MAXP replicas, but only if the task time of one of the tasks assigned to it exceeds a pfe-defined value (α% of the cycle time),
(ix) the zoning constraints are defined for the assembly process - ZP is the set of task pairs that must be assigned to the same workstation and ZN is the set of task pairs that cannot be performed in the same workstation.

Denoting the controllable variables as follows,

$$x_{ik} = \begin{cases} 1, \text{if task i is assigned to workstation k} \\ 0, \text{otherwise} \end{cases} \quad \text{and} \quad r_k = \begin{cases} 1, \text{if workstation k may be 'parallelised'} \\ 0, \text{otherwise} \end{cases}$$

the assembly line balancing problem with parallel workstations and zoning constraints that minimises the number of workstations and balances the workloads across stations, for a given cycle time, can be modelled as the mixed integer linear program shown in Figure 1.

The first term of the objective function (1) minimises the index of the workstation to which the last task is assigned, thus minimising the number of workstations. The second term balances the workload across the workstations, where S' is the actual number of workstations required to meet the demand in the assembly line (S' = k: x_{Nk}=1), s_k is the idle time for workstation k and BDT is the sum of the idle time for all workstations ($BDT = \sum_{k=1}^{S'} s_k$). This term is usually called 'smoothness index', SI, (Moodie and Young, 1965). The value of SI varies between a maximum of 1, when the balance delay time is equal to the idle time of one of the workstations and a minimum of zero when the balance delay time is equally distributed by all the workstations in the line. So, the best solution is the one that minimises the number of workstations and, for that minimum number, distributes tasks into the workstations as evenly as possible.

The constraints can be interpreted as follows:

(2) constraints ensuring that each task is assigned to only one station of the station interval,
(3) constraints ensuring that no successor of a task is assigned to an earlier station than that task,
(4) compatibility zoning constraints,
(5) incompatibility zoning constraints,

(6) set of constraints ensuring that: (i) each workstation time capacity is not exceeded (s_k is a slack variable representing the idle time for station k) (ii) the maximum number of replicas of a workstation is not exceeded and (iii) only workstations where the processing time of the tasks assigned to it exceeds a certain proportion (α%) of the cycle time can be duplicated,
(7) set of integrality constraints.

$$\min Z = \sum_{k=1}^{S} k.x_{Nk} + \frac{S'}{S'-1}\sum_{k=1}^{S'}\left(\frac{s_k}{BDT} - \frac{1}{S'}\right)^2 \quad (1)$$

$$\text{Subject to } \sum_{k=1}^{S} x_{ik} = 1 \qquad i = 1,...,N \quad (2)$$

$$\sum_{k=1}^{S} k.x_{ak} - \sum_{k=1}^{S} k.x_{kb} \le 0 \qquad a \in N; b \in F_a \quad (3)$$

$$\sum_{k=1}^{S} k.x_{ak} - \sum_{k=1}^{S} k.x_{kb} = 0 \qquad (a,b) \in ZP \quad (4)$$

$$x_{ak} + x_{bk} \le 1 \qquad (a,b) \in ZN; k = 1,...,S \quad (5)$$

$$\sum_{i=1}^{N} t_i.x_{ik} + s_k = C[1 + r_k(MAXP - 1)] \qquad k = 1,...,S \quad (6a)$$

$$r_k \le \sum_{i:t_i \ge \ldots} x_{ik} \qquad k = 1,...,S; 0 \le \ldots \le 100 \quad (6b)$$

$$\mathcal{M}.r_k \ge \sum_{i:t_i \ge \ldots} x_{ik} \qquad k = 1,...,S; 0 \le \ldots \le 100 \quad (6c)$$

$\mathcal{M}$ is a very large positive integer

$$x_{ik} \in \{0,1\} \qquad k = 1,...,S; i = 1,...,N \quad (7a)$$

$$r_k \in \{0,1\} \qquad k = 1,...,S \quad (7b)$$

Figure 1. Mathematical programming model for the ALBP1 with parallel workstations and zoning constraints.

The high number of constraints and binary variables of the proposed model makes it impossible to solve by optimising methods, at least for real world problems. Although the number of variables could be reduced by defining, for each task, an earliest and latest station (see Scholl, 1999), the ensuing problem would remain too complex to be solved optimally. Thus, a simulated annealing procedure, presented in the next sections, was developed to tackle the problem.

3. SIMULATED ANNEALING FOR THE ASSEMBLY LINE BALANCING PROBLEM WITH PARALLEL WORKSTATIONS

3.1 General simulated annealing algorithm

Kirkpatrick, Gelatt and Vecchi (1983) initially presented the simulated annealing algorithm, which attempts to solve hard combinatorial optimisation problems through controlled randomisation. Since then the algorithm has been applied to many optimisation problems in a wide variety of areas, including the assembly line balancing problem (Suresh and Sahu (1994); Heinrici (1994); McMullen and Frazier (1998)).
The simulated annealing algorithm evolves from an initial solution for the problem, S_0, which is used as the first current solution, S. A control parameter, T, is set to an initial 'temperature' value, T_0, and is systematically decreased according to an annealing schedule, which defines: (i) a temperature reducing function and (ii) the length of each temperature level, L, that determines the number of solutions generated at a certain temperature. At each temperature level and as the temperature decreases, neighbouring solutions to the current solution are found. If a neighbouring solution, S_n, is as good or better than the current solution ($f(S_n) \le f(S)$), the neighbouring solution becomes the new current solution. If the neighbouring solution is worse than the current solution ($f(S_n) > f(S)$), the neighbouring solution may still replace the current solution with a certain probability, $p = e^{-\Delta/T}$ (where $\Delta = f(S_n) - f(S)$). The most important characteristic of the simulated annealing algorithm is then the possibility of accepting worst solutions, which can allow it to escape from local minima.

Nonetheless, the performance of the algorithm depends on the definition of the several control parameters (annealing schedule):
(i) The initial temperature (T_0) should be high enough so that in the first iteration of the algorithm the probability of accepting worst solutions is, at least, 80% (Kirkpatrick et al, 1983).
(ii) The most commonly used temperature reducing function is geometric: $T_i = a_i.T_{i-1}$ ($a_i < 1$ and constant). Typically, $0.8 \le a_i \le 0.99$. (Eglese, 1990)
(iii) The length of each temperature level, L, determines the number of solutions generated at a certain temperature, T.
(iv) The stopping criterion defines when the system has attained a desired energy level. Some of the most common criteria are based on:
- the total number of solutions generated;
- the temperature at which the desired energy level is attained (freezing temperature);
- the acceptance ratio (ratio between the number of solutions accepted and the number of solutions generated).

Naturally each of these control parameters must be refined according to the specific problem on hand. Two other important issues that need to be defined when adapting this general algorithm to a specific problem are the procedures to generate both the initial solution and the neighbouring solutions. The details of the proposed implementation of the simulated annealing algorithm to the assembly line balancing problem with parallel workstations and zoning constraints are presented in the next section. In addition to the above references further in-depth information on the simulated annealing algorithm can be found in Laarhoven and Aarts (1987).

3.2 Implementation of the proposed simulated annealing algorithm

The main characteristics incorporated in the proposed simulated annealing procedure for the assembly line balancing problem with parallel workstations are as follows:
(i) the goal is to minimise the number of workstations for a given cycle time,
(ii) allows the decision-maker to define an upper bound on the number of replicas of a workstation,
(iii) allows for the inclusion of zoning constraints,
(iv) allows the decision-maker to define the minimum task time (expressed as percentage of the cycle time) required for the workstation performing that task to be replicated (this time will be termed minimum 'parallelisation' time - MPT). By default this value is 100% of the cycle time, which means that only workstations performing tasks whose duration is larger that the cycle time can be duplicated.

3.2.1 Initial Solution

The initial solution is obtained using a version of the Rank Positional Weight (RPW) heuristic proposed by Helgeson and Birnie (1961). The positional weight of a task is the cumulative task time associated with itself and its successors. Tasks are assigned to the lowest numbered feasible workstation by decreasing order of their positional weight. In the original version of the RPW heuristic the cumulative duration of the tasks in a workstation cannot exceed the cycle time (hence the concept of feasible workstation) and thus does not account for parallel workstations. The version of the RPW heuristic used to obtain the initial solution for the simulated annealing procedure redefines the concept of feasible workstation: if a workstation performs a task with a duration larger than MPT its time capacity is C x MAXP, otherwise is C.
The implemented version of the RPW heuristic also checks if the task to be assigned is not incompatible with any of the tasks already allocated to the workstation. The implemented procedure also merges tasks that need to be processed in the same workstation previously, so that they are treated as only one task.

3.2.2 Criteria to evaluate the solutions

Two criteria are used to evaluate the solutions generated by the proposed simulated annealing. The first one is the balance delay time, which is the sum of the idle times for all the stations in the assembly line. The balance delay time is given by:

$$BDT = \sum_{i=1}^{S'} IT_i = \sum_{i=1}^{S'} (SP_i.C - \sum_{j=1}^{SO_i} t_j) \quad (8)$$

where, for the solution to be evaluated, IT_i is the idle time for station i, S' is the number of workstations, SO_i is the number of tasks assigned to workstation i and SP_i is the number of replicas of workstation i. When the balance delay time is minimised, so is the number of workstations in the assembly line. This criterion is then equivalent to the first term of the objective function of the mathematical programming model and it was used to ease the programming effort.
The second criterion balances the workloads across the stations and uses the smoothness index, as stated in the mathematical programming model.

Since, in the practice, the main goal is to minimise the number of workstations, the best solution presented by the heuristic is the one with the minimum number of workstations and, for that number of workstations, with the lowest value of the smoothness index.

3.2.3 Neighbouring solutions

A neighbouring solution can be generated by one of the following actions: (i) swap two tasks in different workstations or (ii) transfer a task to another workstation. The tasks to be swapped, as well as the task and the workstation for the transfer are randomly chosen. For any of these actions to result in a new neighbouring solution the precedence, zoning and workstations time capacity constraints must be fulfilled. When this is not the case, a new task or transfer must be attempted.

Some transfer movements are responsible for the elimination of workstations, having a great contribution on the minimisation of the balance delay time. Swap procedures, on the other hand, allow a better balance of workloads between the workstations. Therefore, the probability of performing a transfer procedure must be higher than for the swap procedure. Probabilities of 75% and 25% respectively were chosen for the test problems bellow, although the user can set different values.

3.2.4 Annealing schedule

The following control parameters were used to implement the algorithm:

(i) Initial temperature (T_0): The computational experience showed that the balance delay time never changed by more than 10% between two neighbouring solutions. So, for an initial temperature of 50 it is guaranteed that at least 80% of the inferior solutions are accepted.

(ii) Temperature reduction function: The geometric function with a temperature reduction factor of 0.9 (T_i=0.9T_{i-1}) was used.

(iii) The length of each temperature level (L): A dominant factor on the computational effort associated to the solution of the problem is the number of tasks (N). So, in order to restrict the computational effort to the first order of the dominant factor, the number of solutions searched at each temperature level was set to K.N, where K is a user defined constant (K=1 is the value suggested by default).

(iv) Stopping criteria: Two alternative criteria were set. In the first one, a freezing temperature of 10 is set, which means that 16 temperature levels are used ($T_0.a_i^{15}$=50.(0.9^{15})=10.29). In the second one, it is admitted that, if in five consecutive temperature levels 85% of the generated solutions are rejected, then the probability of replacing the best solution found is very small and the procedure is then terminated.

Upon termination of the simulated annealing procedure the best solution found is the one with the lowest value for the balance delay time and, consequently, with fewest workstations. However, other solutions with the same number of workstation and higher values of SI might have been found. As the SI might not be the only relevant criterion for the decision-maker to evaluate the solutions, all the solutions with the same number of workstations as the best solution found are stored and presented to the decision-maker.

3.3 Numerical illustration

In this section, a numerical example to demonstrate the proposed simulated annealing procedure is presented. The precedence relationships for the assembly process are shown in the precedence graph in Figure 2, where each node represents a task. The task number is shown inside the circle representing the node and the corresponding task time (in time units - t.u.) is shown above the circle. The line is to be balanced for a cycle time of 9 t.u., taking into account the following constraints: (i) tasks 11 and 13 cannot be performed in the same workstation, (ii) the maximum number of replicas of a workstation is two, and (iii) only workstations performing tasks with a processing time greater than the cycle time can be duplicated (MPT=C).

The initial solution is determined by the modified version of the RPW heuristic. The positional weights for each task are shown, by decreasing order, in Table 1. The assignment sequence of the tasks to the workstations is depicted in Table 2. The first column (Candidate tasks) shows the tasks that in each step can be assigned, because they satisfy both the precedence and zoning constraints and have not yet been assigned. The candidate task with the highest positional weight is selected for assignment (task assigned column). The third and fourth column shows the workstation, and corresponding replica, to which the task is assigned. The time available at each replica after the task assignment is presented in the fifth column.

Table 3 indicates the resulting initial assignment of tasks to workstations. This initial solution has 13 workstations (with a total of 16 replicas), a balance delay time of 16 t.u. and a smoothness index of 14%. After a number of swap and transfer movements, starting from the initial solution, the current solution (named solution A) has a task assignment shown in Table 4. This solution has 13 workstations and a balance delay time of 16 t.u., like the initial solution, and a smoothness index of 25%.

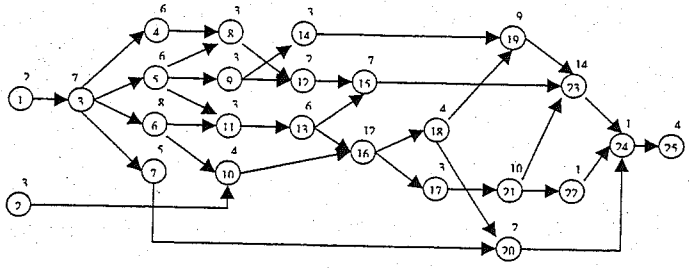

Figure 2. Precedence graph for the numerical example.

Table 1. Positional weight of the tasks.

Task	Successors	Positional weight	Task	Successors	Positional weight
1	3,4,5,6,7,8,9,10,11,12,13,14,15, 16,17,18,19,20,21,22,23,24,25	125	8	12,15,23,24,25	31
3	4,5,6,7,8,9,10,11,12,13,14,15, 16,17,18,19,20,21,22,23,24,25	123	14	19,23,24,25	31
5	8,9,12,14,15,19,23,24,25	93	21	22,24,25	30
6	10,11,13,15,16,17,18,19,20,21, 22,23,24,25	88	12	15,23,24,25	28
11	13,15,16,17,18,19,20,21,22,23, 24,25	76	19	23,24,25	28
13	15,16,17,18,19,20,21,22,23,24, 25	73	15	23,24,25	26
2	10,16,17,18,19,20,21,22,23,24, 25	67	23	24,25	19
10	16,17,18,19,20,21,22,23,24,25	64	7	20,24,25	12
16	17,18,19,20,21,22,23,24,25	60	20	24,25	7
9	14,19,23,24,25	43	22	24,25	6
4	8,12,15,23,24,25	37	24	25	5
18	19,20,23,24,25	34	25	-	4
17	21,22,24,25	33			

Table 2. Sequence of the assignment of tasks.

Candidate tasks	Task assigned	Workstation	Replica	Available time	Obs.	Candidate tasks	Task assigned	Workstation	Replica	Available time	Obs.
1,2	1	1	1	7		7,8,21	21	7	1	0	(iii)
2,3	3	1	1	0					2	1	
2,4,5,6,7	5	2	1	3		7,8,22	22	7	2	0	
2,4,6,7,9	2	2	1	0		7,8	8	8	1	6	
4,6,7,9	6	3	1	1	(i)	7,12,14	14	8	1	3	
4,7,9,10,11	11	4	1	6		7,12,19	12	8	1	1	(i).
4,7,9,10,13	13	4	1	0	(ii)	7,15,19	19	9	1	0	
4,7,9,10,13	10	4	1	2	(i)	7,15	15	10	1	2	(i)
4,7,9,13,16	13	5	1	3		7,23	23	11	1	0	(iii)
4,7,9,16	16	5	1	0	(iii)				2	5	
			2	0		7	7	12	1	4	
4,7,9,17,18	9	6	1	6		20	20	12	1	2	
4,7,17,18	4	6	1	0		24	24	12	1	1	(i)
7,8,17,18	18	7	1	5		25	25	13	1	5	
7,8,17	17	7	1	2							

Observations: (i) All candidate tasks have processing times greater than the available time. Close workstation.
(ii) The task cannot be assigned to the workstation due to zoning constraints.
(iii) The task has a processing time greater than the cycle time. A second replica is required

Table 3. Task assignment for the initial solution.

Workstation	Number of replicas	Tasks assigned	Idle time	Workstation	Number of replicas	Tasks assigned	Idle time
1	1	1,3	0	8	1	8,12,14	1
2	1	2,5	0	9	1	19	0
3	1	6	1	10	1	15	2
4	1	10,11	2	11	2	23	4
5	2	13,16	0	12	1	7,20,24	1
6	1	4,9	0	13	1	25	5
7	2	17,18,21,22	0				

Table 4. Task assignment for solution A.

Workstation	Number of replicas	Tasks assigned	Idle time	Workstation	Number of replicas	Tasks assigned	Idle time
1	1	1,3	0	8	2	17,18,21,22	0
2	1	5,9	0	9	1	12,15	0
3	1	6	1	10	1	19	0
4	1	4,11	0	11	2	23	4
5	1	2,13	0	12	1	20,24	6
6	1	7,10	0	13	1	25	5
7	2	8,14,16	0				

Starting from solution A the algorithm is able to reduce the number of workstations. The transfer of task 23 to workstation 12 is possible since precedence, zoning and capacity constraints are not violated. This way workstation 11 is eliminated. The corresponding solution (solution B) is shown in Table 5. It has 12 workstations (with a total of 15 replicas), a balance delay time of 7 t.u. and a smoothness index of 51%. The high value of the smoothness index is due to the fact that the balance delay time is not equally distributed by the different workstations. Workstation 12 presents itself 71% of the balance delay time while most of the other workstations have a null idle time.

Table 5. Task assignment for solution B.

Workstation	Number of replicas	Tasks assigned	Idle time	Workstation	Number of replicas	Tasks assigned	Idle time
1	1	1,3	0	7	2	8,14,16	0
2	1	5,9	0	8	2	17,18,21,22	0
3	1	6	1	9	1	12,15	0
4	1	4,11	0	10	1	19	0
5	1	2,13	0	11	2	20,23,24	1
6	1	7,10	0	12	1	25	5

Once the number of workstations is reduced, the algorithm tries to improve the value of the smoothness index through swap and transfer movements. The best solution found, using the annealing schedule previously defined, is presented in Table 6. This solution has 12 workstations and a smoothness index of 38%.

Table 6. Task assignment for the best solution.

Workstation	Number of replicas	Tasks assigned	Idle time	Workstation	Number of replicas	Tasks assigned	Idle time
1	1	1,3	0	7	2	8,14,16	0
2	1	5,9	0	8	2	17,18,21,22	0
3	1	6	1	9	1	12,15	0
4	1	4,11	0	10	1	19	0
5	1	2,13	0	11	2	20,23	2
6	1	7,10	0	12	1	24,25	4

In this example the first priority was to minimise the number of workstations and then, for the minimum number of workstations, minimise the smoothness index. If a higher priority was given to the smoothness index then the assembly line would present a higher number of workstations, but that is not the usual case.

4. COMPUTATIONAL EXPERIENCE

The heuristic was coded in Visual C++ on a 233 MHz Pentium II computer. The purpose of the proposed heuristic is to solve assembly line balancing problems with special characteristics like zoning constraints and parallel workstations, in such a way that the user is able to control the 'parallelisation' process, as described earlier. It is not possible to directly compare the heuristic with others reported in the literature as none have the same underlying characteristics. Nevertheless, in order to test the performance of the proposed heuristic, a series of comparative tests were carried out by adapting the heuristic to the conditions under which a series of benchmark problems were originally set. The first test problems proposed by Tonge (1961) and Scholl (1993) are for the SALBP without parallel workstations and optimal solutions are available for them (in this case the heuristic was run setting MAXP=1).

The test problem proposed by Tonge (1961) has seventy tasks, and was executed for 17 different cycle times. For each cycle time ten runs of the proposed simulated annealing procedure were performed. The procedure found 12 optimal solutions, and for the other 5 is marginally worse than the optimal. For each of the ten test runs there was no variability with regards to the number of workstations found for the best solution. In sight of these results, and taking into account the underlying conditions to the problem, the performance of the proposed heuristic can be considered good.

Scholl (1993) proposed a challenging data set that contains very different problem structures. For the 168 instances analysed the optimal solution is known for 161. The solutions obtained by the proposed heuristic for this data set are slightly inferior to the optimal solution for most cases (14% in the worst case), but the procedure manages to find the optimal solution in 76 instances.

The only control on the 'parallelisation' process that the procedures proposed by Buxey (1974) and Schofield (1979) allow is by limiting the number of replicas for the workstations. By making MPT=0 in the proposed procedure it is possible to compare its performance for the test problems used by those authors. The comparative results are shown in Table 7. One can see that the proposed simulated annealing procedure performs very well in these test conditions.

Table 7. Comparison of results for the test problems proposed by Buxey (1974) and Schofield (1979).

Problem		Original solution		Proposed heuristic solution	
		Workstations	%BDT	Workstation	%BDT
Buxey	1	9	0%	10	3,7%
	2	6	0%	6	0%
	3	13	4,14%	13	2,16%
	4	13	4,14%	13	1,44%
	5	13	4,14%	13	1,16%
	6	8	1,23%	8	1,23%
Schofield		14		13	

In order to evaluate the real performance of the heuristic, a set of assembly line problems with special characteristics was generated and a lower bound for the problem, LB_p was derived, based on the following set of assumptions:(i) the maximum number of replicas per workstation is two (MAXP=2), (ii) a workstation can be duplicated only if the task time of one of the tasks assigned to it exceeds the cycle time (α=100%, MPT=C), and (iii) the task time of the longest task does not exceed twice the cycle time (t_{max}<=2C). The following steps are required to compute LB_p :

Step 1:Classification of the tasks according to the corresponding task time, as shown in Table 8.

Table 8. Classification of the tasks to compute LB_p.

Task type	Task time	Task type	Task time	Task type	Task time
A	$5/3C < t_A \leq 2C$	E	$1/3C < t_E \leq 2/3C$	H	$t_H = 2/3C$
B	$4/3C < t_B \leq 5/3C$	F	$t_F = 5/3C$	I	$t_I = 1/3C$
C	$C < t_C \leq 4/3C$	G	$t_G = 4/3C$	J	$t_J < 1/3C$
D	$2/3C < t_D \leq C$				

Step 2: Compute LB'.

LB' is the main term of LB_p. It was derived from on one of the lower bounds, LB_3, defined by Scholl (1999) for the SALPB and adapted to the parallel workstations problem. LB' is computed as follows:

$$LB' = \left\lceil \left(2.(n_A + n_B + n_C) + y.(n_D - n_C) + \frac{1}{2}w.(n_E - n_B) + \frac{5}{3}n_F + \frac{4}{3}n_G + \frac{2}{3}n_H + \frac{1}{3}n_I \right) \right\rceil \quad (9)$$

where n_X is the number of tasks of type X (X=A,...,J), y equals 1 if n_D-n_C>0 or zero otherwise and w equals 1 if n_E-n_B>0 or zero otherwise. The reasoning for this computation is as follows.

The workstations in which long tasks, of types A, B or C, are performed need to be duplicated. As two long tasks cannot share the same workstation, because the value of MAXP would be exceeded, a lower bound for the overall number of workstations (including replicas) is twice the number of tasks of types A, B and C. Each task of type D can be combined with a task of type C in a duplicated workstation, however if there are not enough duplicated workstations of type C to accommodate the tasks of type D, each of these remaining tasks will require a workstation. The same reasoning applies to tasks of type E, that is, two tasks of type E require a single workstation, but can also be combined with a duplicated workstation performing tasks of type B. Finally, the tasks of types F, G, H and I have a fixed task time and so they occupy a fraction of a workstation corresponding to the ratio between their task time and the cycle time.

Step 3: Compute Δ

A second term is added to LB' to compute the value of LB_p. This term adds up the number of workstation needed to process tasks of type J, which in most real problems account for a large proportion of the workstations. Because these tasks can easily be included in workstation that perform tasks of the other types, it is necessary to verify if, after filling up these workstation, there are tasks of type J remaining to open new workstations. The minimum number of workstation required to perform tasks of type J, after filling up the remaining capacity of the workstation assigned to other task types is then given by:

$$\Delta = \left\lceil \left[\sum_{i \in J} t_i - \left(LB'.C - \sum_{i \notin J} t_i \right) \right] / C \right\rceil \qquad (10)$$

Step 4: Compute $LB_p = LB' + \Delta$.

The heuristic was tested on set of 21 problems, whose main characteristics are shown on Table 9, namely, the number of tasks (N) and the cycle time (C). The precedence diagrams for these problems are from Scholl (1993), except for the one with N=25 which is the one from the example in section 3.3. The cycle time and the task times for each problem were generated taking into acount the different task types that might be present in the assembly process. Also shown in Table 9, for each test problem, are: (i) the number of workstations found by the heuristic procedure (Sol.), (ii) the corresponding lower bound computed as explained above and (iii) the deviation between the solution found and the lower bound (Dev.).

Table 9. Computational results for the test problems

Problem	N	C	LB_p	Sol.	Dev.	Problem	N	C	LB_p	Sol.	Dev.	Problem	N	C	LB_p	Sol.	Dev.
1	25	10	19	19	0%	8	30	12	20	21	5%	15	35	15	18	20	10%
2	25	12	19	20	5%	9	30	15	19	21	10%	16	45	10	26	29	10%
3	25	15	18	20	10%	10	32	10	22	24	8%	17	45	12	28	28	0%
4	28	10	20	20	0%	11	32	12	21	23	9%	18	45	15	31	33	6%
5	28	12	19	19	0%	12	32	15	20	21	5%	19	70	10	43	45	4%
6	28	15	21	21	0%	13	35	10	23	25	8%	20	70	12	37	39	5%
7	30	10	20	20	0%	14	35	12	22	23	4%	21	70	15	29	29	0%

Taking into consideration that the proposed lower bound does not account for either precedence or zoning constraints and that the higher deviation between the solutions found and the lower bound is 10%, one can conclude that the heuristic performs very well for this set of problems.

5. CONCLUSIONS

In this paper a new mathematical programming model for the assembly line balancing problem with parallel workstations was presented. The model simultaneously minimises the number of workstations, for a given cycle time, and smoothes the workload among them. Besides taking into account the usual zoning and precedence constraints, the model allows the user to control the way by which the workstations are 'parallelised', either by limiting the number of replicas allowed for each workstation, as by defining a minimum task time that triggers the replication of the workstations.

Due to the model complexity a simulated annealing procedure was proposed to solve it. The comparative performance of the proposed procedure with respect to others available in the literature shows that it renders acceptable results.

The extension of the simulated annealing procedure to the mixed-model line balancing problems with parallel workstations should be a future field of research.

6. REFERENCES

1. Bard, J.F. (1989). Assembly Line Balancing with Parallel Workstations and Dead Time. International Journal of Production Research, 27: 1005-1018.

2. Baybars, I. (1986). An Efficient Heuristic Method for the Simple Assembly Line Balancing Problem. International Journal of Production Research, 24: 149-166.

3. Buxey, G. M. (1974). Assembly Line Balancing with Multiple Stations. Management Science, 20: 1010-1021.

4. Daganzo, C. F. and Blumenfeld, D. E. (1994). Assembly System Design Principles and Tradeoffs. International Journal of Production Research, 32: 669-681.

5. Eglese, R. W. (1990). Simulated Annealing: A tool for Operational Research. European Journal of Operational Research, 46: 271-281.

6. Heinrici, A. (1993). A Comparison between Simulated Annealing and Tabu Search with an Example from the Production Planning. In Operations Research Proceedings (Ed.: Derigs, U., Salomon, M. and Tijms, H.C.). Springer, Berlin, pp. 498-503.

7. Helgeson, W. B. and Birnie, D.P. (1961). Assembly Line Balancing using the Ranked Positional Technique. Journal of Industrial Engineering, 1: 394-398.

8. Johnson, R. V. (1983). A Branch and Bound Algorithm for Assembly Line Balancing Problems with Formulation Irregularities. International Journal of Production Research, 19: 277-287.

9. Kirkpatrick S., Gelatt Jr., C. D. and Vecchi, M. P. (1983). Optimization by Simulated Annealing. Science, 220: 671-680.

10. Laarhoven, P. J. M. and Aarts, E. H. L. (1987). Simulated Annealing: Theory and Practice. Kluwer Academic Publishers, Dordrecht, The Netherlands.

11. McMullen, P. R. and Frazier, G. V. (1998). Using Simulated Annealing to solve a Multiobjective Line Balancing Problem with Parallel Workstations. International Journal of Production Research, 36: 2717-2741.

12. Moodie, C. L. and Young, H. H. (1965). A Heuristic Method of Assembly Line Balancing for Assumptions of Constant or Variable Work Element Times. Journal of Industrial Engineering, 16: 23-29.

13. Pinto, P. A., Dannenbring, D. G. and Khumawala, B. M. (1975). A Branch and Bound Algorithm for Assembly Line Balancing with Paralleling. International Journal of Production Research, 13: 183-196.

14. Pinto, P. A., Dannenbring, D. G. and Khumawala, B. M. (1981). Branch and Bound and Heuristic Procedures for Assembly Line Balancing with Paralleling of Stations. International Journal of Production Research, 19: 565-576.

15. Salveson, M. E. (1955). The Assembly Line Balancing Problem. Journal of Industrial Engineering, 6: 18-25.

16. Sarker, B. R. and Shantikhumar, J. G. (1983). A Generalized Approach for Serial or Parallel Line Balancing. International Journal of Production Research, 21: 109-133.

17. Schofield, N. A. (1979). Assembly Line Balancing and the Application of Computer Techniques. Computers and Industrial Engineering, 3: 53-69.

18. Scholl, A. (1993). Data of Assembly Line Balancing Problems. Schriften zur Quantitativen Betriebswirtschaftslehre 16/93, TH Darmstadt.

19. Scholl, A. (1999). Balancing and Sequencing of Assembly Lines. Physica-Verlag, Heidelberg.

20. Suresh, G. and Sahu, S. (1994). Stochastic Assembly Line Balancing using Simulated Annealing. International Journal of Production Research, 32: 1801-1810.

21. Tonge, F. M. (1961). A Heuristic Program for Assembly Line Balancing. Prentice-Hall, Englewood Cliffs.

International Journal of Industrial Engineering, 8(3), 241-250, 2001.

ANALYSIS OF INTERVAL CENSORED DATA FROM HIGHLY FRACTIONATED EXPERIMENTS

Abdul H. Chowdhury and Nasser S. Fard

Department of Mechanical, Industrial and Manufacturing Engineering
Northeastern University, Boston, MA 02115
E-mail: nfard@coe.neu.edu

In most cases, a highly fractionated life testing experiment involves a large number of factors for a small number of runs. Typical ANOVA assumes that the observed lifetimes in a highly fractionated experiment (HFE) are normally distributed and are not subject to censoring. This paper uses the design of experiment (DOE) technique to design and analyze HFE with interval-censored response data for reliability improvement. The paper also present an alternative method for analyzing interval-censored data from HFE based on regression imputation technique. After imputing censored data, the mean and the rank transformation of the response data are analyzed to deal with the interval-censoring problem. This paper provides experimental plan in analyzing interval-censored data for reliability improvement. The choice of experimental design, interpretation and the role of interactions are discussed. The method is demonstrated by examining the heat exchanger experiment data where lifetimes are measured in several time intervals.

Significance: Many engineering experiments may encounter unforeseen events, which could be terminated prior to their predetermined termination. This paper presents a methodology for imputing censored data in a highly fractionated experiment.

Keywords: Fractionated Experiment, Censored Data, Predicted Values and Mean Ranks.

(Received 28 January 2000; Accepted in revised form 2 March 2001)

1. INTRODUCTION

In many life-testing experiments, accumulation of the exact failure times is often too costly or unfeasible. An alternative and often-feasible approach would require a periodical inspection of test units until all units under the study fail. Interval-censored data are obtained when failures are observed only at inspection times. Then a failure interval consists of an upper and a lower bound. When a unit fails at its first inspection, it is equivalent to a left-censored data, whereas if a unit does not fail by the last inspection, it is equivalent to a right-censored data. The endpoint of an upper bound of an interval is infinity (∞). Interval-censored data does not provide the actual failure times.

The design of experiment (DOE) provides a framework to design and analyze an experiment to determine the quantitative effect of input variables that are deliberately changed on measured response Daniels (1938), Zelen (1959), Box (1955), and Box, Hunter & Hunter (1978). A growing body of researchers has emphasized the DOE ideas for reliability improvement, for instance, Taguchi (1986, 1987), Sullivan (1984, 1987), Hahn, Morgan, & Schmee (1981), Hamada & Wu (1991, 1992), and Lu & Unal (1994). This paper provides classes for experimental plans in analyzing interval-censored data for reliability improvement. The choice of experimental design, interpretation and the role of interactions are discussed briefly.

In this paper, a method is proposed for analyzing interval-censored response data in an unreplicated factorial experiment, since interval censoring causes possible abnormalities in the data set. This method imputes the censored data followed by an analysis of the mean and the rank transformation of the response data. The method is demonstrated by examining a data set of heat exchanger experiment where lifetimes are measured in several time intervals and consequently generates interval-censored data in a highly fractionated experiment (HFE).

1.2 DOE Procedures and Methods for Analyzing Censored Data from HFE

A full factorial experiment investigates all possible combinations for all factor levels. In this design, for k factors and at n levels, the total number of combinations is n^k. Usually designs with more than *2* levels provide excessive runs for analysis. Thus, it is not recommended to use this design unless there are quite a few factors to be examined. Designs with more than *2* levels are referred to as multi-level design, for instance mixed level design, 3^{k-p} design with k factors and p generators, etc.