International Journal of Production Research

ISSN: 0020-7543 (Print) 1366-588X (Online) Journal homepage: http://www.tandfonline.com/loi/tprs20

Balancing of U-type assembly systems using simulated annealing

E. Erel , I Sabuncuoglu & B. A. Aksu

INT. J. PROD. RES., 2001, VOL. 39, NO. 13, 3003–3015

Balancing of U-type assembly systems using simulated annealing

E. EREL†*, I. SABUNCUOGLU‡ and B. A. AKSU‡

The paper presents a new simulated annealing (SA)-based algorithm for the assembly line-balancing problem with a U-type configuration. The proposed algorithm employs an intelligent mechanism to search a large solution space. U-type assembly systems are becoming increasingly popular in today's modern production environments since they are more general than the traditional assembly systems. In these systems, tasks are to be allocated into stations by moving forward and backward through the precedence diagram in contrast to a typical forward move in the traditional assembly systems. The performance of the algorithm is measured by solving a large number of benchmark problems available in the literature. The results of the computational experiments indicate that the proposed SA-based algorithm performs quite effectively. It also yields the optimal solution for most problem instances. Future research directions and a comprehensive bibliography are also provided here.

1. Introduction

The first assembly line is credited to Henry Ford in 1915 after which they have been widely used in various production systems. This recognition of assembly lines is attributable to the high productivity levels achieved in mass production environments. Until 1955, only trial-and-error methods have been used to design assembly lines; Salveson (1955) is the first researcher who presented a formal mathematical analysis of the problem. Bowman (1960) proposed two LP formulations. Since then, an immense amount of literature has accumulated in this area, ranging from exact algorithms (DP, LP, IP) to various heuristics.

Line balancing is the process of allocating a set of tasks (the smallest indivisible portions of the assembly operation) to an ordered sequence of stations in such a way that some performance measures (e.g. cycle time, number of stations) are optimized subject to the precedence relations among the tasks. Task time and station time are defined as the duration to perform a task and the actual amount of work, in time units, assigned to a station, respectively. The maximum station time constitutes the cycle time. An example will be used to clarify these concepts. Consider Jackson's 11-task problem depicted in figure 1. The numbers in and below the nodes represent the tasks and the associated task times, respectively. The directed arrow between tasks i and j implies that task i is a predecessor of task j. Assuming a cycle time of 10 (this might be imposed to achieve a certain production rate), a solution of the line-balancing problem is depicted in figure 2a. Note that the tasks are allocated to six stations with station times of 9, 9, 8, 7, 8 and 4, respectively.

Revision received December 2000.

† Department of Management, Bilkent University, Ankara 06533, Turkey.
‡ Department of Industrial Engineering, Bilkent University, Ankara 06533, Turkey.
* To whom correspondence should be addressed. e-mail: Erel@bilkent.edu.tr

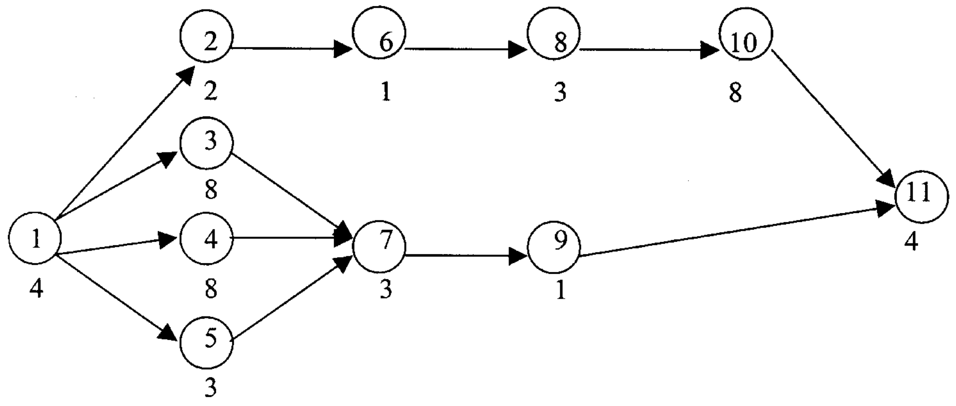

Figure 1. Precedence diagram and the task times of the example problem.

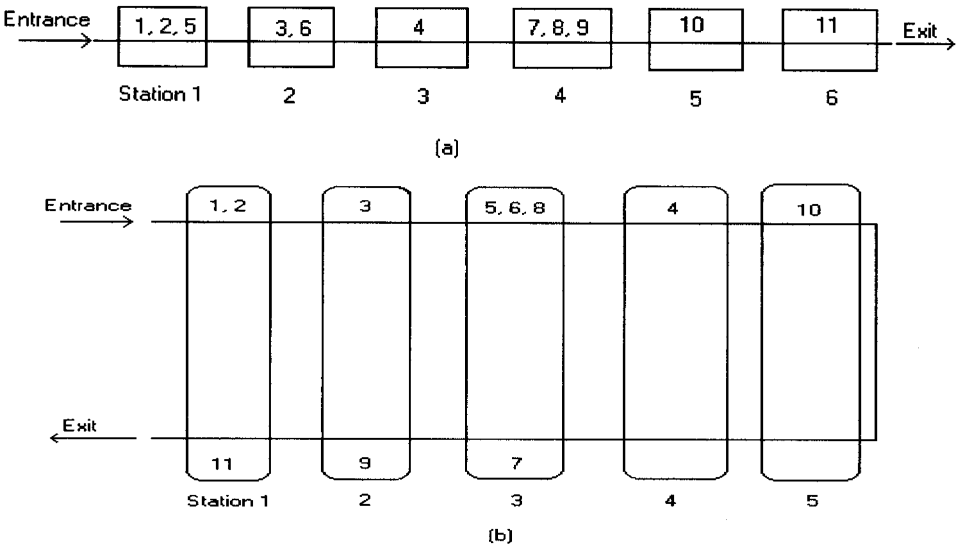

Figure 2. Schematic view of (a) straight line and (b) U-line configurations for the example problem.

As stated above, this problem has been extensively studied in the literature for decades. These studies, together with a taxonomy of the problem, are summarized in several survey papers (Baybars 1986, Ghosh and Gagnon 1989, Erel and Sarin 1998).

In 1993, Monden identified more general configurations (U- and S-type topologies) of the traditional straight-line system. He observed these configurations in Japanese manufacturing companies. The important characteristic of these new systems is that multiskilled (cross-trained) workers perform various tasks of different stations along the production line. This feature allows assigning tasks both from the beginning and the end of the precedence diagram to the same stations. In terms of the solution of the line-balancing problem, this implies that the solution of the U-type configuration dominates the solution of the traditional straight-line configuration due to the reduced precedence relations (i.e. the number of stations on a U-type configuration is less than or equal to the number required on a straight line). On the other hand, the problem of assigning tasks on a U-type configuration is more complex than a straight line due to the much larger search space.

Figure 2b depicts a solution of the previous example with a U-type configuration. In this solution, the worker in station 1 first completes tasks 1 and 2 of an item and then completes task 11 of the previous item. As noted, workers in a U-type configuration may work on two different items within the same cycle. Note also that this solution has five stations (one less than the straight-line configuration) with station times of 10, 9, 10, 8 and 8, respectively.

The need for U-type assembly configuration arises from attempts to improve productivity with greater flexibility. Monden (1993) discusses these issues in the context of the Toyota Production System. In general, the following advantages of the U-type configuration can be stated: (1) a quick response to changes in the environment (machine breakdowns, worker absenteeism, etc.) due to the possibility of reallocating the cross-trained workers, (2) the ease to adopt to changes in cycle time, because of the high potential of rebalancing the line with a new cycle time, (3) a high level of participation of workers to improve production process, (4) the flexibility of adding/removing workers (rebalancing lines), and, most importantly, (5) the number of stations required by a U-line is never greater than the one by a traditional line. In spite of these benefits, U-lines present some operational difficulties in scheduling the movements of workers, dispatching jobs, material handling activities and WIP control. Moreover, the line-balancing problem of an U-line is much more complicated than the traditional line due to the increased search space.

This paper develops a new simulated annealing-based algorithm to minimize the number of stations in such a U-type assembly line. An assembly line will be called a U-line hereafter.

The rest of the paper is as follows. The relevant literature on the U-type line-balancing problem and simulated annealing are given in Section 2. This is followed by the structure of the proposed algorithm and the example in Section 3. Computational results are presented in Section 4. Conclusions and future research directions are given in Section 5.

2. Literature review

The literature on the U-lines is sparse and new relative to the traditional straight lines. The research on U-lines can be classified into two groups: line balancing (ULB) and production flow lines. In the former group, the researchers study the problem of balancing U-type assembly systems to minimize either cycle time or the number of stations. In the latter group, the emphasis is on identifying the important design factors and their effects on the performance of U-type flow lines. Since the latter case is not within the scope here, see Nakade and Ohno (1997, 1999), Nakade *et al.* (1997) and Miltenburg (2000).

As stated above, Monden (1993) brought U-lines to the attention of the scientific community. The first ULB study in the literature was by Miltenburg and Wijngaard (1994), who developed a DP formulation for the single-model U-line to minimize the number of stations. The proposed algorithm was used to solve problems with up to 11 tasks. The authors presented a Ranked Positional Weight Technique (RPWT)-based heuristic for larger size problems (111-task problems have been solved). Later, Miltenburg and Sparling (1995) developed three exact algorithms to solve the ULB

Authors	Methodology	Problem specs	Solved problems	Objectives
Miltenburg and Wijngaard (1994)	DP formulation RPWT-based heuristic	single model	up to 11 tasks up to 111 tasks	number of stations
Miltenburg and Sparling (1995)	DP-based exact algorithm depth-first and breadth-first B&B	single model	up to 40 tasks	number of stations
Sparling and Miltenburg (1998)	heuristic	mixed model	up to 25 tasks	number of stations
Urban (1998)	IP formulation	single model	up to 45 tasks	number of stations
Miltenburg (1998)	DP-based exact algorithm	U-line facility with several individual U-lines	individual U-lines with up to 22 tasks	number of stations and idle time in a single station
Sparling (1998)	heuristic	U-line facility with several individual U-lines	up to nine U-lines with up to 18 tasks in each U-line	number of stations
Scholl and Klein (1999)	B&B-based heuristic	single model	up to 297 tasks	number of stations, cycle time, and both

Table 1. Summary of the work conducted on U-type assembly lines.

problem. The first was based on a reaching DP formulation, whereas the other two were breadth- and depth-first branch-and-bound (B&B) algorithms. Their computational experiments indicated that the B&B-based algorithms were more efficient (taking less time to reach optimal solutions) than the DP-based algorithm. On average, the breadth-first algorithm required less computational time than the depth-first algorithm. However, depth-first algorithms either found the optimal solution quickly or took a very long time. The authors solved problems with up to 40 tasks.

Later, Urban (1998) developed an IP formulation for the same problem and managed to solve problems with up to 45 tasks. Eliminating the variables representing assignment of tasks to certain stations reduced the size of the model.

Scholl and Klein (1999) solved the ULB problem by the B&B procedure, which was adapted from their previous algorithm called SALOME developed for the straight ALB problem. The new procedure, called ULINO (U-line optimizer), is basically a depth-first B&B algorithm with various bounding and dominance rules to minimize the number of stations, or cycle time, or both. In this study, the authors reported the results of some preliminary computational tests on problems with up to 297 tasks; the procedure yielded promising results especially for the objective of minimizing number of stations.

Mixed-model U-lines were studied by Sparling and Miltenburg (1998). They developed a heuristic procedure for the U-line by which different products were assembled simultaneously. Their approximate solution algorithm that merges each model's precedence graph into a single precedence graph solved problems with up to 25 tasks.

Miltenburg (1998) proposed a reaching DP formulation for a U-line facility that consisted of numerous U-lines connected by multiline stations. The DP is capable of balancing with any number of U-lines provided that individual U-lines do not have more than 22 tasks and do not have wide, sparse, precedence graphs. Sparling (1998) developed heuristic solution algorithms for a U-line facility consisting of individual U-lines operating at the same cycle time and connected with multiline stations. Travel time between tasks and U-lines were also considered. The author solved problems with up to nine individual U-lines which have up to 18 tasks each.

All these studies have demonstrated that the ULB is an important problem for today's modern assembly systems. There are several exact methods for their solution. However, as can be noted in table 1, up to 45-task problems can be optimally solved due to the complex nature of the problem. Hence, as also stated by Miltenburg (1998), computationally more effective heuristic procedures are needed for larger-sized problems. In the present paper, such a procedure is proposed that can easily handle more than 100-task problems.

3. Proposed SA-based algorithm

3.1. *Motivation*

As stated before (Baybars 1986, Miltenburg and Sparling 1995, Miltenburg 1998, Sparling and Miltenburg 1998, Sabuncuoglu *et al.* 1999), ALB problems are combinatorial problems. Since the exact methods (DP and B&B) can handle only a limited size, heuristics are generally recommended for large-size problems.

Although numerous heuristics with different characteristics have been developed for the line-balancing problems (for a comprehensive review of the ALB problems, see Erel and Sarin 1998), there are only two heuristic procedures proposed for the U-type ALB problems in the literature. The first is based on the well-known RPWT of

Helgeson and Birnie (1961) and is adapted for the U-type ALB problem by Miltenburg and Wijngaard (1994). This is a single-pass greedy heuristic and, hence, it may easily get stuck at a local optimum. The second heuristic is based on the B&B technique (Scholl and Klein 1999). It is an optimum-seeking procedure (a depth-first multi-pass method) that attempts to optimize the number of stations, cycle time, or both.

In the present paper, a new heuristic is proposed that utilizes the concepts of simulated annealing (SA) within an intelligent search mechanism. The initial motivation was to see the performance of a fairly straightforward application of SA in such a combinatorial problem environment. However, during the algorithmic construction, a new way of searching the solution space was devised that, together with SA, resulted in better solutions. There are mainly two components of the proposed algorithm: a solution generator module and an SA module. As will be discussed in detail below, the first module generates a new solution by relaxing the cycle time constraint. The role of SA is to take over this solution and obtain feasible task assignments. In contrast to the existing SA applications in the literature where the goal is to optimize a certain performance measure, SA was employed to reconstruct the feasibility. Hence, the application is different than the traditional applications of SA.

In general, SA is a metaheuristic, which is already proven to be very effective in dealing with combinatorial problems (Kuik and Salomon 1990, van Laarhoven *et al.* 1992, Suresh and Sahu 1994). To the best of our knowledge, the present study is the first SA application to the U-type ALB problem. The other application in the line-balancing literature (Suresh and Sahu 1994) is for the stochastic ALB problem. The SA module in the proposed algorithm differs from theirs with respect to the objective function, neighbour generation and internal structure.

As a background information about SA, we note the following. SA is a random search technique that has received considerable interest from researchers in various fields; it has been especially applied to the problems that are difficult to solve (i.e. combinatorial problems). This search technique is first proposed by Kirkpatrick *et al.* (1983) for optimization problems by inspiring from the physical annealing of metals. SA simulates the change in the energy of the system subject to a cooling process until a frozen state is reached. The states of the physical system and the energy of a state correspond to the solutions and the objective function value of the optimization problem, respectively. The details and a lengthy discussion of SA are beyond the scope of this paper, but see Johnson *et al.* (1989, 1991) and Reeves (1993).

3.2. *Overview of the algorithm*

The structure of the proposed algorithm is illustrated in figure 3. It consists of two main parts: the solution generator and SA module. The solution generator is a mechanism that generates a new (and better) solution from the old solution by relaxing the cycle time constraint. Then, the SA module takes the solution and reallocates the tasks to stations with the objective of minimizing the maximum station time; hence, it attempts to construct a feasible allocation. These two stages of the algorithm are invoked consequently until a stopping criterion is met; the algorithm stops either when a prespecified time limit is exceeded, or when the minimum number of stations is obtained, or when a feasible allocation cannot be reconstructed by the SA module (in that case it returns to the previous feasible solution).

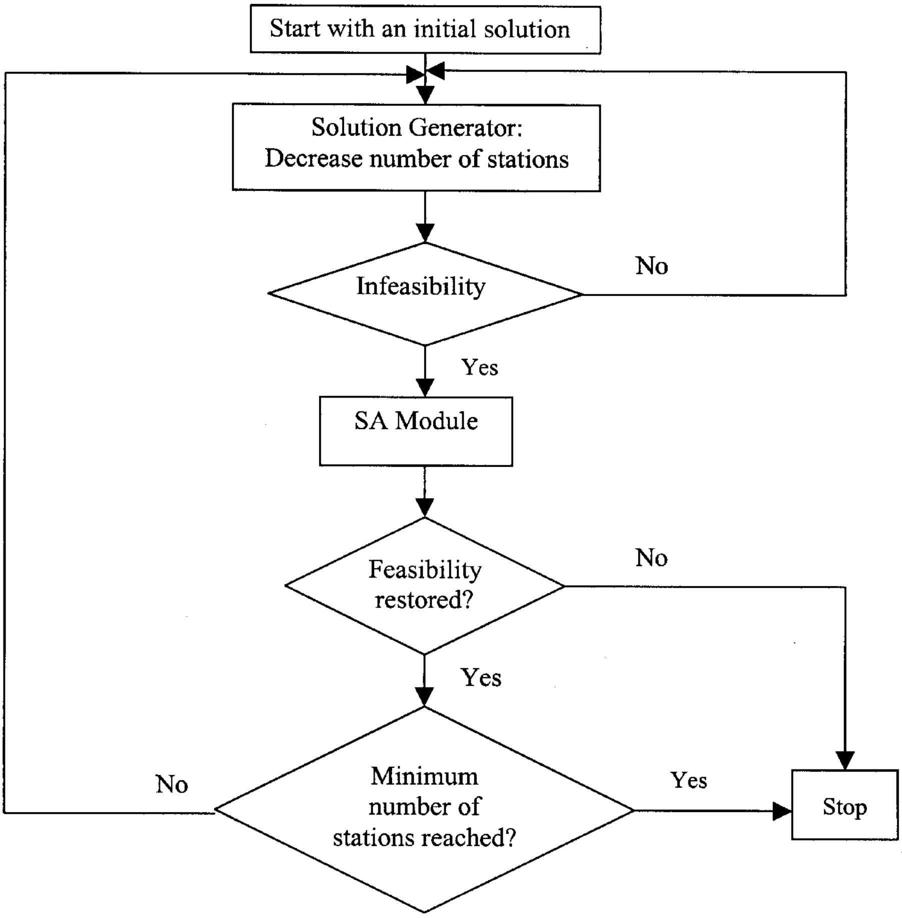
Start with an initial solution
Solution Generator: Decrease number of stations
Infeasibility
No
Yes
SA Module
Feasibility restored?
No
Yes
No
Minimum number of stations reached?
Yes
Stop

Figure 3. Structure of the proposed algorithm.

3.2.1. *Solution generator*

Initially, tasks are assigned sequentially to separate stations (i.e. number of stations equals to number of tasks). This initial solution is obviously feasible but not a good one due to the excessive number of stations. Hence, one has to reduce the number of stations by reallocating tasks. Given an initial or a previous solution (i.e. current allocation of the tasks to the stations satisfying all the precedence and cycle time constraints), the solution generator checks each pair of subsequent stations and combines one of the two adjacent stations that yield the minimum total station time. As a result, it finds a new assembly line design with one less station than the previous one. This process is repeated until the station time of the combined station exceeds the cycle time (i.e. leading to infeasibility). When an infeasibility is detected, as will be explained below, the SA module is invoked to find a feasible allocation by minimizing the maximum station time. If a feasible allocation is found, the generator further decreases the line length. Otherwise, the algorithm terminates with the previous feasible solution. To achieve feasibility, SA reassigns the tasks in the combined station obtained by the solution generator to other stations by minimizing the maximum station time.

3.2.2. *SA module*

The goal of the SA module is to reconstruct feasibility for the solution produced by the solution generator. In a typical SA application, the analyst determines the initial temperature, temperature reduction, cost function and neighbour generation scheme. These are explained below.

3.2.2.1. *Neighbour generation.* Generating a neighbour of the current solution is an important decision since it directly affects the efficiency of the algorithm. In the proposed algorithm, two operators are implemented: interchange (SWAP) of two tasks and inserting (INSERT) a task to a different station. In the former case, two randomly selected tasks from different stations are simply exchanged. In the latter case, a randomly selected task is inserted into the station with the minimum station time. In each case, feasibility of the resulting change (in terms of satisfying the precedence relations) is checked; the process is repeated until a feasible move is obtained. In order to determine the SWAP and INSERT probabilities, pilot experiments (the 83-task problem of Arcus is solved for 10 different seeds) are conducted. Based on the results (considering both the average CPU time and solution quality), SWAP and INSERT probabilities (p) are set to 0.5 (table 2).

3.2.2.2. *Temperature setting.* Setting the initial temperature along with the temperature reduction scheme is another important decision in SA applications. Initial temperature is usually set high to reduce the possibility of getting trapped at a local optimum. Then, this temperature is reduced gradually to achieve better solutions.

Again, we conducted pilot simulation experiments with 10 different seeds on the 83-task problem of Arcus to determine the initial temperature and temperature reduction function. Based on the results, we have set the initial temperature to 80 and used a linear temperature reduction function of 0.80 T. As seen in tables 3 and 4, these levels yield the minimum average CPU time. It should be stated here that similar results for initial temperature and temperature reduction function have been obtained from the pilot runs conducted on other test problems.

SWAP probability	Average CPU time (s)	Maximum CPU time (s)	Minimum CPU time (s)
0.9	31.40	96.28	6.65
0.7	9.45	14.56	4.73
0.5	8.15	17.30	4.89
0.3	10.55	24.72	5.22
0.1	22.80	36.25	12.64

Table 2. Computation times of the algorithm for the 83-task problem for different SWAP probabilities.

Initial temperature	Average CPU time (s)	Maximum CPU time (s)	Minimum CPU time (s)
60	9.20	17.47	4.17
80	9.06	21.25	3.97
100	12.07	26.76	5.11
120	13.88	47.68	3.52
140	13.89	30.74	6.54
160	14.43	26.53	7.30
180	15.95	31.56	7.14
200	16.13	41.22	9.24

Table 3. Data about the simulated annealing runs to obtain initial temperature.

Temperature reduction function	Average CPU time (s)	Maximum CPU time (s)	Minimum CPU time (s)
0.70	9.21	20.87	2.14
0.75	9.22	19.77	4.34
0.80	9.20	23.72	3.68
0.85	9.90	16.20	5.20
0.90	10.66	22.52	3.40
0.95	10.99	20.83	2.14

Table 4. Data about the simulated annealing runs to obtain temperature reduction function.

3.2.2.3. *Cost function.* We have used the maximum station time as the cost of the SA module. The reason for using this cost measure is that the feasibility is achieved indirectly by minimizing the maximum station time. By this way, we aim to eliminate the infeasibility caused by the station (with the maximum station time) exceeding the cycle time.

The SA module terminates when no change is observed for three consecutive trials at a specific temperature. An illustrated example is given below to clarify further the logic behind the algorithm.

3.2.2.4. *Example.* Consider Jackson's 11-task problem depicted in figure 1 (discussed above). The solution of the example by the proposed algorithm can be seen in figure 4. The boxes and the numbers in them represent the stations and tasks assigned to these stations, respectively. The numbers below the stations represent the associated station times. Initially, each task is assigned to a different station that results in a line design with 11 stations (i.e. one task for each station). Then, the solution generator takes this feasible, but certainly not the optimal solution, and searches for the station pair with the minimum total station time. In the example, stations 5 and 6 are combined into a single station, since these stations yield the minimum total station time ($3 + 1 = 4$). Then stations 7 and 8 are combined and this process is repeated until a station time exceeds the cycle time (i.e. infeasibility is encountered). In the example, infeasibility is first detected when the line length is reduced to six stations (station 4 has a station time $= 11$). The SA module takes this infeasible solution and attempts to reconstruct feasibility. In the

Initial solution:

1	2	3	4	5	6	7	8	9	10	11
4	2	8	8	3	1	3	3	1	8	4

Solution generator: Iteration 1

1	2	3	4	5, 6	7	8	9	10	11
4	2	8	8	4	3	3	1	8	4

Solution generator: Iteration 2

1	2	3	4	5, 6	7	8, 9	10	11
4	2	8	8	4	3	4	8	4

Solution generator: Iteration 3

1, 2	3	4	5, 6	7	8, 9	10	11
6	8	8	4	3	4	8	4

Solution generator: Iteration 4

1, 2	3	4	5, 6, 7	8, 9	10	11
6	8	8	7	4	8	4

Solution generator: Iteration 5

1, 2	3	4	5, 6, 7, 8,9	10	11
6	8	8	11	8	4

SA module result:

1, 2	3	4	5, 6, 8	10	7, 9
					11
6	8	8	7	8	8

Solution generator: Iteration 6

1, 2, 3	4	5, 6, 8	10	7, 9
				11
14	8	7	8	7

SA module result:

1		2, 3	4, 6	5, 7
11	10, 9			8
8	9	10	9	9

Figure 4. Solution of the example problem.

example, the tasks are allocated/reallocated to the stations by the SA module yielding a feasible solution with a maximum station time = 8. Next, the solution generator is again invoked to further reduce the line length. In this second iteration, the generator yields a five-station design with a maximum station time = 14. Then the SA module reallocates the tasks to achieve feasibility. In the example, a feasible solution is obtained with a maximum station time = 10. At the end of this stage, the algorithm terminates, since the theoretical minimum number of stations is reached ($\lceil 45/10 \rceil = 5$). Note that a lower bound on the number of stations is equal to ⌈summation of task times/cycle time⌉, where $\lceil x \rceil$ is the smallest integer $> x$. The resulting design has a U-type configuration.

4. Computational results

The performance of the proposed algorithm is measured with the well-known benchmark problems in the literature. These problems have been previously tested by different researchers in the U-type ALB literature. They were classified here into two data sets. The first was created from earlier studies in the area (Miltenburg and Wijngaard 1994, Urban 1998, Hoffmann 1990). The second set is relatively new and has been developed by Scholl and Klein (1999) as the most complex data set for this problem.

We classify the first data set into three categories: small (up to 11 tasks), medium (up to 45 tasks) and large (above 45 tasks). Optimal solutions of the small-size problems are already given by Miltenburg and Wijngaard (1994) using a DP formulation. For the medium-size problems, Urban (1998) offers the optimal solution using an IP formulation. For the large-size problems, optimal solutions have not yet been reported in the literature; only the RPWT results (not necessarily optimal) are available for these problems.

The results indicated that the proposed algorithm finds the optimal solution in 20 out of 21 instances. On the other hand, the RPWT finds the optimal solution in 17 instances. For the medium-size problems, the proposed algorithm finds the optimal solution of all the 23 instances, whereas RPWT finds the optimal solution for only 11. Optimal U-line solutions of large-size problems are not available in the literature with only the optimal straight-line solutions being reported there. Hence, the proposed algorithm is compared with the results of the RPWT. It is interesting to note that both the proposed algorithm and the RPWT yield the same results on these instances. Finally, as can be expected, the U-line solutions by these heuristics are better than or identical to the optimal straight-line solutions (better solutions are obtained in eight out of 62 instances, yielding the same for the remaining ones).

Another experimentation conducted in the ULB literature is due to Scholl and Klein (1999b). Here, the B&B-based heuristic procedure, ULINO (U-Line Optimizer), was applied to the ULB problem. ULINO is adapted from the authors' previous algorithm called SALOME (Simple Assembly Line Balancing Optimization Method) for the ALB problem (Scholl and Klein 1999a). In these studies, Scholl and Klein formed a new data set of 168 problems with varying problem sizes ranging from 25 to 297 tasks. (The data set is available at: http://www.bwl.tu-darmstadt.de/bwl3/) Scholl and Klein (1999b: 732) compared their algorithm with the RPWT of Helgeson and Birnie (1961), as modified by Miltenburg and Wijngaard (1994), and found that the performance of ULINO was superior to RPWT.

At the second stage of the present computational experiments, we applied our algorithm to this new data set and compared the results with those of ULINO. The results indicate that the algorithm finds the optimal solution in 112 out of 168 problems. In 13 problems of the remaining 56 instances, the algorithm yielded the solution within the optimal solution range provided by Scholl and Klein on their website. Together with these instances, we obtained 125 optimal solutions out of 168 instances. When each instance was analysed, it was noted that the proposed algorithm generates better solutions than ULINO for eight instances. In five instances, the results of ULINO are provided by intervals and the present solutions fall into these intervals. In the remaining 155 instances both algorithms produce the same results. Later, during the process of preparing this paper, Scholl and Klein (1999b) reported the improved results with their algorithm (150 instances are solved to optimality). The computation time requirements of the both algorithms are comparable. They basically solved reasonably complex and large size problems (297 tasks) in seconds. Besides, these problems are not operational problems (like a scheduling problem) that require frequent and on-line solutions. Hence, the solution time by itself is not a major issue in assessing the quality of algorithms. In terms of the solution quality, it is concluded that the algorithm proposed here is a competent solution procedure of the ULB problem in the literature.

5. Conclusions and further research directions

This paper studied the U-type assembly line-balancing problem and an SA-based heuristic algorithm was developed. The performance of the proposed algorithm is measured against two well-known optimum-seeking algorithms (DP- and IP-based algorithms) and the heuristic procedures. The results of the computational experiments on various test problems indicate that the proposed method performs well. Besides, the computational requirements are not high; the average computation time for an 83-task problem (i.e. large-size problem) is about 31.4 s on a Pentium II 266 kHz machine with 32 MB RAM. The computational success of the SA-based algorithm can be attributed to the intelligent search of a larger search space.

As noted previously, most of the line-balancing algorithms in the literature are developed for solving the ALB problem (there are only a few algorithms for the ULB problem). However, as indicated by many researchers (Miltenburg and Sparling 1995, Urban 1988, Scholl and Klein 1999b), the recent trend in the applications of JIT principles has led to a widespread use of U-lines in assembly systems. In this context, the present study offers a new and efficient solution procedure for the need of the industry.

We can also list the following research directions: first, the proposed methodology can be applied to more difficult problems such as mixed/multi-model assembly systems, assembly lines with stochastic task performance times and different U-line configurations. Second, it would be interesting to use other metaheuristics (e.g. genetic algorithms, tabu search) in the U-type assembly line balancing problem. By that way, one can assess the relative effectiveness of the modern search heuristics. Third, it is also important to mention that there is a need to develop exact algorithms that can handle large-size problems. Finally, we believe that this metaheuristic approach can be effectively used to solve the Type II problem (i.e. minimizing cycle time subject to a given number of stations) by iteratively solving a series of Type I problems (i.e. minimizing number of stations given a cycle time).

References

Baybars, I., 1986, A survey of exact algorithms for the simple assembly line balancing problem. *Management Science,* **32**, 909–932.

Bowman, E. H., 1960, Assembly-line balancing by linear programming. *Operations Research,* **8**, 385–389.

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

Ghosh, S. and Gagnon, R. J., 1989, A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems. *International Journal of Production Research,* **27**, 637–670.

Helgeson, W. B. and Birnie, D. P., 1961, Assembly line balancing using the ranked positional weight technique. *Journal of Industrial Engineering,* **12**, 394–398.

Hoffmann, T. R., 1990, Assembly line balancing: a set of challenging problems. *International Journal of Production Research,* **28**, 1807–1815.

Johnson, D. S., Aragon, C. R., McGeoch, L. A. and Schevon, C., 1989, Optimization by simulated annealing: an experimental evaluation, Part I, Graph partitioning. *Operations Research,* **37**, 865–892.

Johnson, D. S., Aragon, C. R., McGeoch, L. A. and Schevon, C., 1991, Optimization by simulated annealing: An experimental evaluation, Part II, Graph coloring and number partitioning. *Operations Research,* **39**, 378–476.

Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P., 1983, Optimization by simulated annealing. *Science,* **220**, 671–680.

Kuik, R. and Salomon, M., 1990, Multi-level lot-sizing problem: evaluation of a simulated annealing heuristic. *European Journal of Operational Research,* **45**, 25–37.

Miltenburg, J., 1998, Balancing U-lines in a multiple U-line facility. *European Journal of Operations Research,* **109**, 1–23.

Miltenburg, J., 2000, The effect of breakdowns on U-shaped production lines. *International Journal of Production Research,* **38**, 353–364.

Miltenburg, J. and Sparling, D., 1995, Optimal solution algorithms for the U-line balancing problem. Working Paper, McMaster University, Hamilton.

Miltenburg, J. and Wijngaard, J., 1994, The U-line balancing problem. *Management Science,* **40**, 1378–1388.

Monden, Y., 1993, *Toyota Production System* (Norcross, GA: Industrial Engineering and Management Press, Institute of Industrial Engineers).

Nakade, K. and Ohno, K., 1997, Stochastic analysis of a U-shaped production line with multiple workers. *Computers and Industrial Engineering,* **33**, 809–812.

Nakade, K. and Ohno, K., 1999, An optimal worker allocation problem for a U-shaped production line. *International Journal of Production Economics,* **60–1**, 353–358.

Nakade, K., Ohno, K. and Shanthikumar, J. G., 1997, Bounds and approximations for cycle times of a U-shaped production line. *Operations Research Letters,* **21**, 191–200.

Reeves, C. R., 1993, *Modern Heuristic Techniques for Combinatorial Optimization Problems* (New York: Wiley).

Sabuncuoglu, I., Erel, E. and Tanyer, M., 1999, Assembly line balancing using genetic algorithms. *Journal of Intelligent Manufacturing*, **11**, 295–310.

Salveson, M. E., 1955, The assembly line balancing problem. *Journal of Industrial Engineering,* **6**, 18–25.

Scholl, A. and Klein, R., 1999a, Balancing assembly lines effectively—a computational comparison. *European Journal of Operational Research,* **114**, 50–58.

Scholl, A. and Klein, R., 1999b, ULINO: optimally balancing U-shaped JIT assembly lines. *International Journal of Production Research,* **37**, 721–736.

Sparling, D., 1998, Balancing just-in-time production units: the N U-line balancing problem. *Information Systems and Operational Research,* **36**, 215–237.

Sparling, D. and Miltenburg, J., 1998, The mixed-model U-line balancing problem. *International Journal of Production Research,* **36**, 485–501.

Suresh, G. and Sahu, S., 1994, Stochastic assembly line balancing using simulated annealing. *International Journal of Production Research,* **32**, 1801–1810.

Urban, T. L., 1998, Note. Optimal balancing of U-shaped assembly lines. *Management Science,* **44**, 738–741.

van Laarhoven, P. J. M., Aarts, E. H. L. and Lenstra, J. K., 1992, Job shop scheduling by simulated annealing. *Operations Research,* **40**, 113–125.