International Journal of Production Research

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

Stochastic assembly line balancing using beam search

E. Erel , I. Sabuncuoglu & H. Sekerci

To cite this article: E. Erel , I. Sabuncuoglu & H. Sekerci (2005) Stochastic assembly line balancing using beam search, International Journal of Production Research, 43:7, 1411-1426, DOI: 10.1080/00207540412331320526

To link to this article: http://dx.doi.org/10.1080/00207540412331320526

Published online: 22 Feb 2007.

International Journal of Production Research,
Vol. 43, No. 7, 1 April 2005, 1411–1426

Taylor & Francis
Taylor & Francis Group

Stochastic assembly line balancing using beam search

E. EREL[†]*, I. SABUNCUOGLU[‡] and H. SEKERCI[‡]

[†]Department of Management and [‡]Department of Industrial Engineering,
Bilkent University, Ankara 06800, Turkey

(*Revision received July 2004*)

This paper presents a beam search-based method for the stochastic assembly line balancing problem in U-lines. The proposed method minimizes total expected cost comprised of total labour cost and total expected incompletion cost. A beam search is an approximate branch and bound method that operates on a search tree. Even though beam search has been used in various problem domains, this is the first application to the assembly line balancing problem. The performance of the proposed method is measured on various test problems. The results of the computational experiments indicate that the average performance of the proposed method is better than the best-known heuristic in the literature for the traditional straight-line problem. Since the proposed method is the first heuristic for the stochastic U-type problem with the total expected cost criterion, we only report its results on the benchmark problems. Future research directions and the related bibliography are also provided in the paper.

Keywords: Assembly line balancing; U-type assembly line; Beam search

1. Introduction

An assembly line consists of a sequence of stations performing a specified set of tasks repeatedly on consecutive items moving along the line. The development of the first assembly line is credited to Henry Ford in 1915, after which they have been widely used in various industrial settings to achieve high productivity levels in mass production environments.

Line balancing is the process of allocating the set of tasks to stations to accomplish an assembly work. Tasks are indivisible elements of the assembly work; the duration to perform a task is called the task time. The amount of work assigned to a station is called the station time; the maximum station time constitutes the cycle time of the line. Note that the production rate is the reciprocal of this cycle time. Several managerial and technological factors may impose restrictions on the order in which tasks are performed; thus, precedence constraints specify the permissible sequences of the tasks. Hence, we can define assembly line balancing problem (ALBP) as 'given a finite set of tasks, each having a task time, and a set of precedence constraints

*Corresponding author. E-mail: erel@bilkent.edu.tr

International Journal of Production Research
ISSN 0020–7543 print/ISSN 1366–588X online © 2005 Taylor & Francis Ltd
http://www.tandf.co.uk/journals
DOI: 10.1080/00207540412331320526

which specify the permissible orderings of the tasks, the problem is to assign the tasks to an ordered sequence of stations with a prespecified cycle time such that the precedence constraints are satisfied and some performance measure is optimized'.

1.1 *Background*

For over forty years following the design of Ford's assembly line, only trial-and-error methods have been used to balance assembly lines. Salveson (1955) is the first researcher who presented a formal mathematical analysis of the problem. Since then, an immense amount of literature has accumulated in this area ranging from exact algorithms to various heuristics. The problem has a finite but extremely large number of feasible solutions and the problem's inherent integer restrictions result in enormous computational and storage difficulties. In fact, the ALBP falls into the NP-hard class of combinatorial optimization problems and numerous research efforts have been directed towards the development of computer efficient approximation algorithms or heuristics (Ghosh and Gagnon 1989). These heuristics include various constructive, backtracking and meta-heuristics; the common characteristic of them is the use of problem-specific knowledge intelligently to reduce the search efforts.

ALBP can be classified into two classes as deterministic and stochastic; the former considers task times as constants whereas the latter considers them as random variables. The stochastic version of the problem is especially important when the majority of the tasks are performed manually so that variation in task times from item to item is inevitable. The stochasticity of task times has been recognized and stated by several authors (Arcus 1966, Kottas and Lau 1973, 1976, 1981, Suresh and Sahu 1994). Most of the procedures developed for the stochastic version are modified extensions of the procedures developed for the deterministic version. However, there are also a few heuristics developed exclusively for the stochastic version. In this paper, we consider the stochastic version since it represents manufacturing environments more realistically although the computational complexity increases significantly.

In the last decade researchers have identified more general configurations of assembly lines in Japanese manufacturing companies; these lines have U- and S-type shapes (we will call them U-lines hereafter). The example in figure 1 will be used to clarify the features of these more general shapes. In figure 1a, Jackson's 11-task problem is depicted; the numbers in and below the nodes represent the tasks and the associated task times, respectively. The directed arc between tasks i and j implies that task i is a predecessor of task j. For a cycle time of 10 (this might be imposed to achieve a certain production rate), a solution to the traditional ALBP is given in figure 1b. Note that the tasks are allocated to five stations with station times of 9, 9, 7, 8, and 6, respectively. A solution of the U-line problem is depicted in figure 1c. Note that the U-line solution requires only four stations and this reduction in number of stations is achieved by allowing workers to process tasks of different items within the cycle. For example, the worker in station 1 first completes tasks 1 and 4 of an item and then completes task 11 of another item. Hence, the number of stations in a U-line constitutes a lower bound for the number of stations in a straight line.

Monden (1993) discusses the advantages of these U-lines as follows: (1) A quick response to changes in the environment due to the possibility of reallocating

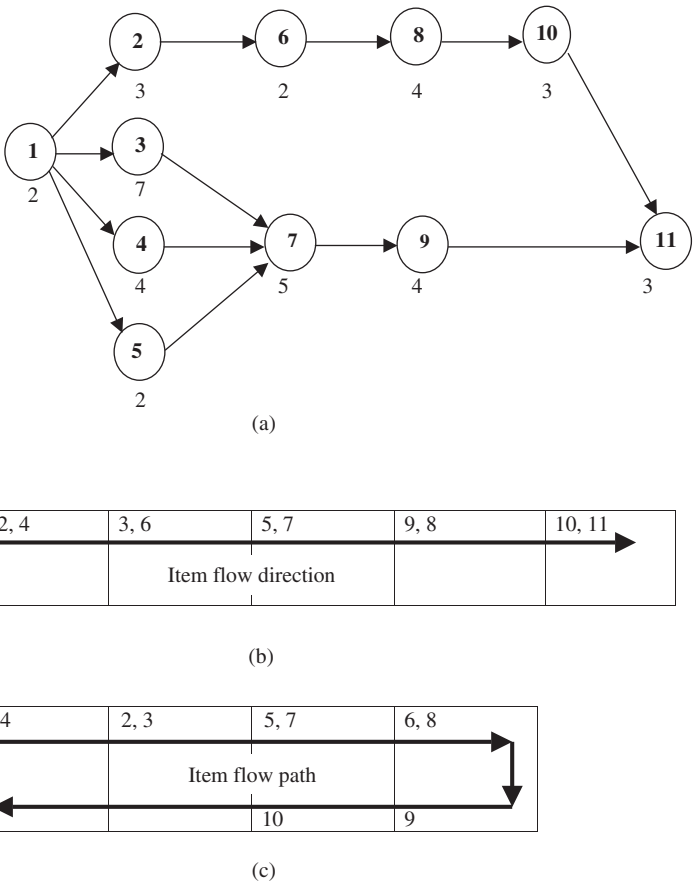


Figure 1. Example problem with straight and U-line solutions. (a) Jackson's 11-task problem (b) Straight line solution (c) U-line solution.

the workers, (2) the ease to adopt to changes in cycle time, (3) a high level of participation of workers to improve production process, (4) the flexibility of adding/removing workers (rebalancing lines), (5) the number of stations in a U-line is lower bound on the number of stations in 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 balancing problem of an U-line is much more complicated than the traditional line due to the increased search space. In this paper, we examine a U-line considering the above advantages of these configurations. Note also that the traditional configuration is a special case of the U-shape configuration.

1.2 *Relevant literature*

In this study we solve the stochastic U-line problem for the objective of minimizing total expected cost comprising total labour cost and total expected incompletion cost. There is no exact or heuristic procedure to solve this problem. The most relevant study to ours is due to Guerriero and Miltenburg (2003) who solve the stochastic U-line problem with a different objective function. The authors developed

an exact recursive algorithm to minimize the sum of number of stations and workload in the last station subject to the constraint that in each station the probability of incompletion is smaller than a pre-specified value. The computational experiments indicate that their exact algorithm is capable of solving problems with up to 25 tasks. The authors suggest heuristics for larger size problems as a further research topic. All the other studies in the literature deal with the deterministic version of the U-line balancing problem. Since some ideas and solution characteristics of the deterministic problem can be extended to the stochastic version, we refer the interested reader to the following papers (Miltenburg and Wijngaard 1994, Sparling and Miltenburg, 1998, Urban 1998, Scholl and Klein 1999, Erel *et al*. 2001).

The rest of the paper is as follows: The detailed description of the proposed method is given in section 2. This is followed by experimental settings in section 3. Computational results are presented in section 4. Finally the conclusions and future research directions are given in section 5.

2. Proposed method

The proposed method to solve the line-balancing problem is a heuristic based on beam search. Beam search is a fast and approximate branch-and-bound method, which operates on a search tree (Sabuncuoğlu and Karabük 1998). It is an adaptation of the branch and bound method in which only some nodes are evaluated. The nodes of the search tree correspond to partial designs (partial solutions). Beam search is a breadth-first search type partial enumeration technique since it progresses level by level without backtracking. It differs from branch-and-bound in the sense that a certain number of paths are selected and the rest is permanently pruned off during the search process. Thus, at any level in the search tree only the promising nodes are kept for further branching and the other nodes are simply ignored. It moves downward from the best β promising nodes at each level until a complete design is obtained. The parameter β is called the beam width.

In order to select the best β nodes, an estimate of the promise of each node is determined. This value can be determined in various ways: One way is to employ a global evaluation function that estimates the minimum total costs of the best solution that can be obtained from the partial solution represented by that node. Another approach would be a one-step evaluation function that employs surrogate measures or priority rules. There is a trade off between these two approaches. One-step evaluation is quick but it may discard good solutions. On the other hand, the global evaluation function can be more accurate but it is computationally more expensive. Regardless of the complexity of global evaluation function, beam search itself has polynomial time complexity because a large part of the search tree is pruned off (Sabuncuoğlu and Karabük 1998). We use beam search not only because of its speed but also because of additional desirable properties such as flexibility of handling various system details, ease of coding the resulting algorithm, and its successful performance in solving difficult optimization problems.

In figure 2, a sample beam search tree is shown. We select the best β number of nodes from the nodes emanating from the root node by comparing the value of the global evaluation function of these nodes. After determining the first beam nodes at level 1, the algorithm is applied to these nodes independently and a partial tree is generated from each of them. Since there are β nodes at any level and it progresses

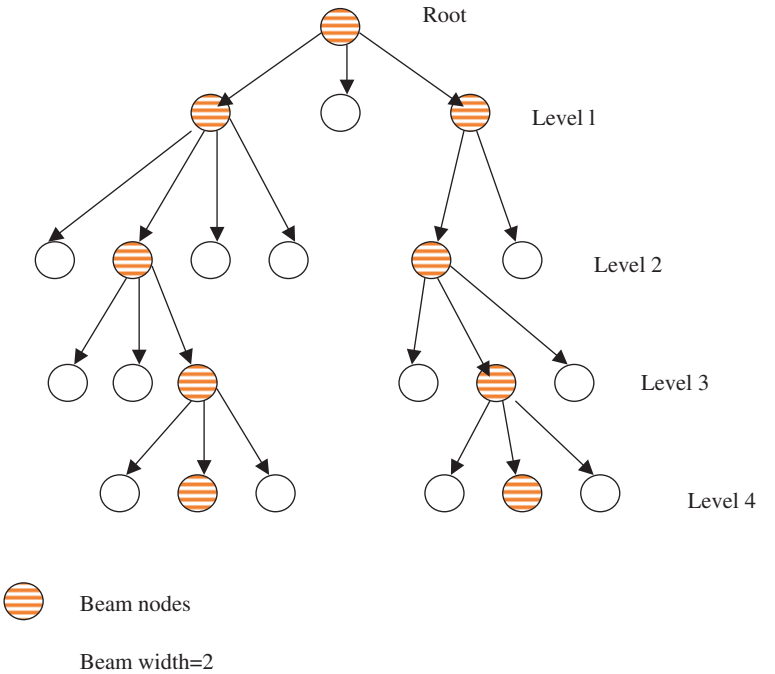


Figure 2. Sample beam search tree.

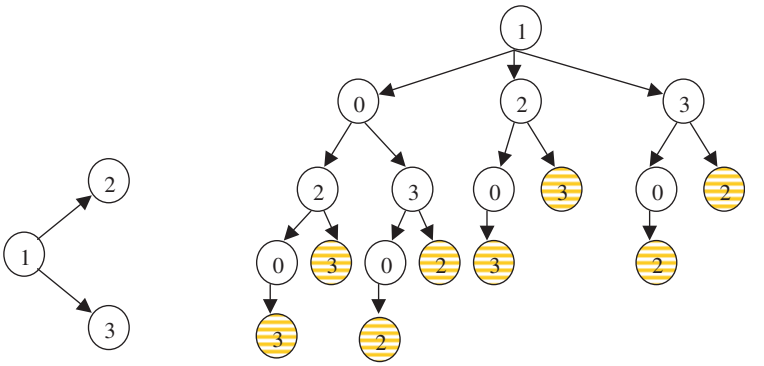
by keeping one descendant only at each beam node, there are β parallel beams resulting in β different solutions at the end.

This search technique was used first by Lowerre (1976), in the artificial intelligence area for the speech recognition problem. Later, Ow and Morton (1988), Sabuncuoğlu and Karabük (1998), and Sabuncuoğlu and Bayiz (2000) applied it to FMS and job shop scheduling problems. To the best of our knowledge, beam search has not been used in the context of assembly line balancing. Thus, to the best of our knowledge, this study is the first application of the beam search methodology to solve the ALBP.

2.1 *Beam search formulation*

When using a beam search algorithm there are two important issues to consider: (1) search tree representation and (2) application of a search methodology.

2.1.1 Search tree representation. In ALBP, each node in the tree corresponds to a partial design. A partial design is an incomplete design where some tasks are allocated to opened stations but there are still tasks to assign and more stations to be opened. In this scheme, an arc between two nodes represents a decision to add a new task to the existing station. The leaf nodes at the end of the tree correspond to complete designs. To facilitate further understanding of the search tree representation, a small precedence diagram (figure 3a) and the corresponding search tree are given in figure 3b. The path 1-2-0-3 in figure 3b indicates that the tasks 1 and 2 are assigned to the first station in the given order and task 3 is assigned to the second



(a) Precedence diagram (b) Beam search tree

Figure 3. A beam search example.

station. The dashed nodes represent the final nodes of their paths. Note that a dummy (task 0) is employed to close a station. These dummy tasks can be assigned whenever needed in the search process.

2.1.2 Search methodology. The second issue in beam search is to determine a search methodology. In the proposed method, all the nodes at any level are globally evaluated to determine the best β promising nodes. The selected nodes become the first nodes of the β parallel beams. Subsequently the descendants of these selected nodes are globally evaluated to select the beam node at each parallel beam. If the number of nodes expanded at the very first level is less than the specified beam width, then all the nodes in the following levels are expanded until the number of nodes at a level is greater than the specified beam width. The procedural form of the proposed beam search method is given as follows:

2.2 *The proposed beam search method*

Step 0 (*initial node generation*): Determine the set of available tasks for assignment considering the precedence relations. These tasks constitute the level 1 nodes. Each node represents a partial design in which the selected task is assigned to the first position in the first station.
Step 1 (*checking the number of nodes*): If number of level 1 nodes is less than the specified beam width (β), then expand nodes by generating further level nodes until the total number of nodes in the last level is greater than the specified beam width. Else go to Step 2.
Step 2 (*completing the partial designs*): The nodes in the search tree represent partial designs. In order to evaluate these partial designs, they must be transformed into complete designs by assigning the remaining tasks via some rules.
Step 3 (*computing global evaluation function*): Compute the global evaluation function for all the nodes and keep the best β.

For each beam node:
Step 4 (node generation): Generate descendants of these beam nodes by considering the set of available tasks for that node. Set of available tasks for a node is determined

by considering all up-to-then assigned tasks for that node from the precedence diagram and choosing the ones with no unassigned predecessors. For the U-line balancing problem, however, a task must have both its predecessors and followers assigned in order to become available for assignment. We also consider a dummy task as a descendant provided that the beam node under consideration has not been assigned in the previous level (closing the current station).
Step 4.1 (*computing global evaluation function*): Compute the global evaluation values of each of these nodes.
Step 4.2 (*selecting beam nodes*): For each beam in the tree, select the node with the lowest global evaluation value (i.e. beam node). Go to Step 5, if there is no task to assign; else go to Step 2.
Step 5 (*selecting the solution design*): Among the beam nodes (complete designs), select the one with the minimum objective value.

2.3 *Heuristics to complete partial designs*

When the search procedure evaluates a node that represents a partial design, it performs two operations: First, the design must be completed by a fast heuristic so that all tasks are assigned to stations. Then, total cost of the completed design is estimated by a heuristic or an exact procedure. This two-step procedure calculates the value of global evaluation function. Note that in most cases it is not possible to give one algebraic expression of the global function. Because in the beam search methodology a global evaluation function is a procedure that estimates the quality of solution from the current node by pruning remaining nodes in the search tree quickly using some heuristic methods. When we reach to the end of the search tree by using these heuristics, the final objective function value of the complete solution becomes the value of global function (i.e. cost of the design) for that particular node. We explain these heuristic procedures as follows.

HEUR 1 (Heuristic for completing partial straight line designs). We use Kottas and Lau's (1973) heuristic and its extension (Kottas and Lau 1981) for completing the partial designs. The key idea is the concept of marginal desirability, which compares the incompletion cost of assigning the task to the current station versus the cost of opening a new station. A task is classified as desirable if the incompletion cost is less than the cost of a new station. This selection process is guided by some rules. In this paper we use the largest incompletion cost selection rule and random selection rule. Largest incompletion cost rule selects the desirable tasks with the largest incompletion cost of the unit assembled. Random rule makes selection randomly.

HEUR 2 (Heuristic for completing partial U-line designs). We modify the Kottas and Lau's (1981) heuristic to complete a partial U-line design. Since in U-lines two units move simultaneously in opposite directions (forward direction in the upper portion of the line and backward direction in the lower portion) at any station, in the assignment process one must consider not only the tasks whose predecessors have already been assigned, but also those tasks whose successors have already been assigned. In other words, tasks whose predecessors have already been assigned are available for assignment in forward direction at a station, whereas tasks whose successors have already been assigned, are also available for assignment in backward

direction. We use the largest incompletion cost rule for forward assignment and the lowest incompletion rule for backward assignment.

HEUR 3 (Heuristic for evaluating complete U-line designs). Rather than the exact procedure for the straight line, a simple heuristic is used for the U-line balancing problem. This heuristic is based on the concepts discussed in Kottas and Lau (1976); the tasks allocated to later parts of task sequence in a station can be incomplete. In order to determine these tasks, the following method is used: A threshold value of $C - \alpha\sqrt{\sigma_{\text{sta}}^2}$ is computed for each station where α is a confidence parameter and σ_{sta}^2 is the station variance calculated as the sum of task time variances of the tasks assigned to the station. We choose α as 2, which implies that the probability of a task being incomplete prior to this point is less than 0.03. Having determined these tasks that are candidate for incompletion, we estimate their contributions to the incompletion cost. This expected cost is approximated by the following relation:

$$\text{Ex. Cost} \approx \sum_{i \in \text{candidates}} \left(P\{\text{First incompletion is on task } i\} * r * (\mu_i + \sum_{j \in F(i)} \mu_j) \right)$$

where r is the off-line completion rate, μ_i is the expected task time of task i, and $F(i)$ is the set of tasks following task i in the precedence diagram.

A task sequence is obtained by appending the tasks assigned in backward direction, if any, to the tasks assigned in forward direction. If a task is at position j in the task sequence, then its incompletion probability is calculated as:

$$P = \phi\left(\frac{C - \sum_{i=1}^{j-1} \mu_i}{\sqrt{\sum_{i=1}^{j-1} \sigma_i^2}} \right) - \phi\left(\frac{C - \sum_{i=1}^{j} \mu_i}{\sqrt{\sum_{i=1}^{j} \sigma_i^2}} \right)$$

where $\phi(x)$ represents the area under standard normal distribution function to the left of point x, C is the cycle time, μ_i and σ_i^2 are the expected task time and variance of task time of task i, respectively.

Even though the heuristics are generally preferred in beam search application, some exact procedures can also be used to calculate the global function. One such example is as follows:

EXACT 1 (Exact procedure for evaluating complete straight line designs). The global evaluation function used to compute expected cost of straight lines is adapted from Kottas and Lau (1976). This method works by identifying all the possible combinations of incomplete tasks, which occur in any unit coming off the line. It is an exact procedure and utilizes the fact that each incompletion combination (IC) associated with a K station line design is uniquely represented by a K-tuple $(n_1, n_2, \ldots, n_K)$ where n_k is the number of tasks assigned to station k which are not completed due to lack of time. A set generation process is used to identify the tasks belonging to each IC represented by the K-tuples as the latter are indexed through all their feasible values.

3. Experimental settings

The input parameters of the proposed method and their values are discussed below.

3.1 *Cycle time*

The theoretical range for the cycle time in a deterministic ALBP is between the maximum task time (M) and the sum of all task times (S). In the stochastic version of the problem, however, it can take any positive value since the objective of total expected cost minimization allows a task incompletion event during the assembly process. In other words, a small cycle time can cause incompletion events and too many stations whereas a large cycle time can result in unacceptably low production rates and trivial problem instances from a computational point of view (i.e. over-simplification of the problem). In the experiments we use three levels for cycle time as follows: M, $(3M + S)/4$, and $(M + S)/2$, respectively. These three levels are selected from the first half of the cycle time range for the deterministic problem.

3.2 *Task time parameters and distributions*

Majority of the researchers in the literature assume normally distributed task times, even though there are a few who use distributions other than normal (for example, Magazine *et al*. (2000) assumed shifted exponential distribution). We also assume that task times are normally distributed random variables with known means and variances. We take standard deterministic test problems from the literature: means of the task times are taken as the deterministic task times of these problems whereas the standard deviation of the task times are generated by multiplying the means by a coefficient of variation (CV) term. Silverman and Carter (1986) have used 0.1 and 0.25 for low and high CV values, respectively. We also use these two levels to set the standard deviations of the task times.

3.3 *Off-line completion rates*

A task becomes incomplete due to two reasons: the actual task time is realized to be too large to get completed within the cycle time, or the task is a follower of another incomplete task so that it cannot be processed. Whatever the reason, we assume that when a task is incomplete, the unit moves down the line with as many of the remaining tasks being completed as possible. The incomplete tasks are completed off the line with some additional costs. The off-line completion rate determines this extra cost of task completion. Assuming that the on-line labour rate is unity, the off-line completion rate can be taken as a multiple of the mean task time. Silverman and Carter (1986) use the levels of 1.5, 5, and 10 for low, medium, and high off-line rates, respectively. Note that a rate of 1.5 implies that the off-line completion cost is 50% higher than the on-line completion cost. We use two levels (1.5 and 5) for the off-line completion rates.

3.4 *Beam width*

Beam width is an important parameter of the proposed heuristic; it limits the size of the search tree. It is reported in the scheduling literature that in most of the beam

search applications, no significant improvement is observed when the beam width is larger than five or six. Since there is no beam search application to ALBP, to determine the beam width we take pilot runs using some selected test problems for different cycle times. We observe that a beam width of two or three usually produces good solutions; hence, we set the beam width to two in order to keep the computational burden of the proposed algorithm at a reasonable level.

4. Computational results

4.1 *Computational results for traditional straight ALBP*

We evaluate the performance of the proposed algorithm by comparing it with the heuristic of Kottas and Lau (1981) on standard test problems; number of tasks ranges from 11 to 70 and total number of instances is 72. The parameters of these problems are obtained from the website of Scholl and Klein (1999) at http://www.bwl.tu-darmstadt.de/bwl3/.

The results of the proposed method and the heuristic of Kottas and Lau (K&L) are presented in table 1 for various problem parameters, including cycle time, incompletion rate, and CV of task times. As can be seen in this table, the proposed method yields better results in 64 out of the 72 instances; the improvement ranges from 0.1% to 24%. In seven instances, K&L heuristic yields better results with the improvement being no more than 0.67%. In the remaining one instance, both methods yield the same solution.

A short description of K&L heuristic is as follows. This procedure attempts to minimize the total expected cost comprised of total labour cost and total expected incompletion cost. Total labour cost is proportional to the number of stations whereas the total expected incompletion cost arises from tasks not being completed within the cycle time. Tasks available for assignment are examined to determine if they are 'desirable' for assignment to the current station. A task is identified as desirable if the expected incompletion cost resulting from assigning it to the current station is less than opening a new station. A task is selected from the desirable-task list according to some criteria, and the procedure is repeated until no task remains for assignment. The computational and storage requirements of the procedure are minimal and several alternative designs can be obtained by varying the selection criteria of the desirable-task list.

In contrast, the proposed method tests all the tasks in the desirable-list one by one to find the most suitable task to assign to the current station. Our computational results indicate that this property leads to the significant improvements (up to 24%) in some of the instances solved. Note that this property also increases the CPU time ranging from a few seconds to 40 minutes. This additional computational burden is reasonable for strategic planning (long-term) problems such as ALBP. But actually solution quality is very important as it affects the operating cost of the system.

4.2 *Computational results for U-line balancing*

In the U-line literature, there is only one study (Guerriero and Miltenburg 2003) which proposes an exact procedure to minimize number of stations subject to the constraint that incompletion probability is smaller than a pre-specified value at each station.

Table 1. Comparison of the proposed method and Kottas and Lau heuristic on straight line problems.

Problem	Cycle time	Off-line coefficient	CV	Beam-search cost*	K&L cost*	Percent improvement
Jackson 11-task	10	1.5	0.15	61.96 (6)	70.64 (7)	13.99
			0.25	67.23 (6)	75.55(7)	12.35
		5	0.15	65.34 (6)	72.09 (7)	10.34
			0.25	88.03 (6)	88.96 (8)	1.06
	15	1.5	0.15	59.13 (3)	60.28 (4)	1.94
			0.25	62.75 (4)	62.83 (4)	0.12
		5	0.15	60.93 (4)	64.59 (4)	6.01
			0.25	70.87 (4)	76.58 (5)	8.05
	20	1.5	0.15	60.70 (3)	60.85 (3)	0.25
			0.25	61.63 (3)	61.83 (3)	0.32
		5	0.15	60.23 (3)	60.23 (3)	0.00
			0.25	63.42 (3)	64.83 (3)	2.22
Mitchell 21-task	20	1.5	0.15	140.43 (7)	140.83 (7)	0.28
			0.25	146.88 (7)	148.44 (7)	1.06
		5	0.15	140.65 (7)	140.95 (7)	0.21
			0.25	165.79 (8)	170.39 (8)	2.77
	30	1.5	0.15	122.57 (4)	123.77 (4)	0.98
			0.25	125.99 (4)	131.74 (4)	4.57
		5	0.15	150.18 (5)	155.58 (5)	3.60
			0.25	151.52 (5)	151.66 (5)	0.10
	40	1.5	0.15	122.24 (3)	126.94 (3)	3.85
			0.25	123.14 (3)	123.80 (3)	0.53
		5	0.15	122.40 (3)	122.72 (3)	0.27
			0.25	160.97 (4)	160.38 (4)	−0.37
Sawyer 30-task	40	1.5	0.15	406.32 (10)	420.62 (10)	3.52
			0.25	467.13 (11)	471.85 (11)	1.01
		5	0.15	441.95 (11)	454.05 (11)	2.74
			0.25	500.23 (12)	508.70 (12)	1.63
	70	1.5	0.15	371.06 (5)	373.95 (5)	0.78
			0.25	384.52 (5)	392.45 (5)	2.06
		5	0.15	423.15 (6)	422.67 (6)	−0.11
			0.25	431.73 (6)	434.59 (6)	0.66
	100	1.5	0.15	404.28 (4)	403.31 (4)	−0.24
			0.25	404.84 (4)	406.16 (4)	0.33
		5	0.15	400.40 (4)	400.53 (4)	0.03
			0.25	407.89 (4)	408.68 (4)	0.19
Kilbridge 45-task	100	1.5	0.15	709.69 (7)	712.60 (7)	0.41
			0.25	708.19 (7)	713.94 (7)	0.81
		5	0.15	705.92 (7)	707.07 (7)	0.16
			0.25	715.21 (7)	719.82 (7)	0.64
	150	1.5	0.15	608.21 (4)	609.95 (4)	0.29
			0.25	756.27 (5)	757.06 (5)	0.10
		5	0.15	755.00 (5)	753.95 (5)	−0.14
			0.25	756.82 (5)	760.98 (5)	0.55
	200	1.5	0.15	603.42 (3)	604.31 (3)	0.15
			0.25	618.06 (3)	619.98 (3)	0.31
		5	0.15	609.60 (3)	611.53 (3)	0.32
			0.25	806.70 (4)	808.88 (4)	0.27

(*continued*)

Table 1. Continued.

Problem	Cycle time	Off-line coefficient	CV	Beam-search cost*	K&L cost*	Percent improvement
Warnecke 58-task	500	1.5	0.15	2005.36 (4)	2017.01 (4)	0.58
			0.25	2008.47 (4)	2021.55 (4)	0.65
		5	0.15	2004.28 (4)	2004.37 (4)	0.00
			0.25	2015.53 (4)	2017.59 (4)	0.10
	600	1.5	0.15	1800.41 (3)	1807.98 (3)	0.42
			0.25	1803.42 (3)	1805.97 (3)	0.14
		5	0.15	1800.31 (3)	1805.64 (3)	0.30
			0.25	1812.15 (3)	1816.77 (3)	0.25
	700	1.5	0.15	2106.24 (3)	2105.22 (3)	−0.05
			0.25	2105.63 (3)	2109.98 (3)	0.21
		5	0.15	2103.09 (3)	2106.59 (3)	0.17
			0.25	2121.40 (3)	2107.20 (3)	−0.67
Tonge 70-task	800	1.5	0.15	4006.04 (5)	4007.36 (5)	0.03
			0.25	4044.19 (5)	4843.51 (6)	19.76
		5	0.15	4030.83 (5)	4091.74 (5)	1.51
			0.25	4174.48 (5)	4297.41 (5)	2.94
	1000	1.5	0.15	4003.69 (4)	4010.00 (4)	0.16
			0.25	4016.54 (4)	4028.21 (4)	0.29
		5	0.15	4050.62 (4)	4095.63 (4)	1.11
			0.25	4100.00 (4)	4136.76 (4)	0.90
	1200	1.5	0.15	4801.93 (4)	4801.32 (4)	−0.01
			0.25	4802.06 (4)	4816.60 (4)	0.30
		5	0.15	3885.45 (3)	4823.26 (4)	24.14
			0.25	4912.99 (4)	4931.00 (4)	0.37

*The values in parentheses indicate the number of stations.

In our study, we minimize expected total costs without imposing any constraint on the incompletion probabilities of the stations. Since there is neither an exact method nor a heuristic procedure available in the literature for the cost-based evaluation of U-type assembly systems, we only report the expected total costs and the associated number of stations. Since the proposed method is tested on the same problems used for the straight-line configuration, we have an opportunity to compare both U- and straight-line solutions on the same problem instances.

In our problem, exact calculation of incompletion probabilities is very difficult since the incompletion probability tree expands rapidly as the problem size increases. Also, two assembly units moving in both directions at stations makes the probability of one unit dependent on the other unit. Thus, the proposed method provides an approximation of the cost. In order to make sure that this estimate is accurate, we develop a simulation model of the system to check if it lies in the confidence intervals (i.e. it is within the desired accuracy level). ARENA 7.0 is used and the output data is collected in steady state for 10 independent replications for each line design. A 95% confidence interval is constructed for the average cost per unit produced by the assembly line. The results verify the fact that cost estimates of the proposed method lie within the 95% confidence intervals.

The overall results presented in table 2 indicate that the U-line configuration reduces the total expected costs in most of the problem instances (57 out of 72 instances). Although the U-line dominates the straight line (due to the fact that the U-type configurations provide more design opportunities than their straight-line

Table 2. Results of the proposed method for the straight line and U-line problems.

Problem	Cycle time	Off-line coefficient	CV	Cost of Straight line*	Cost of U-line*
Jackson 11-task	10	1.5	0.15	61.96 (6)	62.43 (5)
			0.25	67.23 (6)	66.58 (6)
		5	0.15	65.34 (6)	65.33 (6)
			0.25	88.03 (6)	80.08 (7)
	15	1.5	0.15	59.13 (3)	61.04 (4)
			0.25	62.75 (4)	62.50 (4)
		5	0.15	60.93 (4)	60.35 (4)
			0.25	70.87 (4)	75.38 (5)
	20	1.5	0.15	60.70 (3)	60.02 (3)
			0.25	61.63 (3)	60.95 (3)
		5	0.15	60.23 (3)	60.09 (3)
			0.25	63.42 (3)	62.49 (3)
Mitchell 21-task	20	1.5	0.15	140.43 (7)	123.57 (6)
			0.25	146.88 (7)	121.97 (6)
		5	0.15	140.65 (7)	140.88 (7)
			0.25	165.79 (8)	147.45 (7)
	30	1.5	0.15	122.57 (4)	121.44 (4)
			0.25	125.99 (4)	125.01 (4)
		5	0.15	150.18 (5)	126.09 (4)
			0.25	151.52 (5)	150.71 (5)
	40	1.5	0.15	122.24 (3)	120.40 (3)
			0.25	123.14 (3)	122.63 (3)
		5	0.15	122.39 (3)	120.67 (3)
			0.25	160.97 (4)	125.99 (3)
Sawyer 30-task	40	1.5	0.15	406.32 (10)	406.67 (10)
			0.25	467.13 (11)	404.26 (10)
		5	0.15	441.95 (11)	431.51 (10)
			0.25	500.53 (12)	472.12 (11)
	70	1.5	0.15	371.06 (5)	361.69 (5)
			0.25	384.52 (5)	367.33 (5)
		5	0.15	423.15 (6)	421.53 (6)
			0.25	431.73 (6)	426.16 (6)
	100	1.5	0.15	404.28 (4)	402.19 (4)
			0.25	404.84 (4)	403.60 (4)
		5	0.15	400.40 (4)	400.18 (4)
			0.25	407.89 (4)	403.84 (4)
Kilbridge 45-task	100	1.5	0.15	709.69 (7)	618.78 (6)
			0.25	708.19 (7)	708.37 (7)
		5	0.15	705.92 (7)	702.60 (7)
			0.25	715.21 (7)	710.08 (7)
	150	1.5	0.15	608.21 (4)	605.16 (4)
			0.25	756.27 (5)	754.29 (5)
		5	0.15	755.00 (5)	623.02 (4)
			0.25	756.82 (5)	753.77 (5)
	200	1.5	0.15	603.42 (3)	602.51 (3)
			0.25	618.06 (3)	611.63 (3)
		5	0.15	609.60 (3)	602.43 (3)
			0.25	806.70 (4)	800.73 (4)

(*continued*)

Table 2. Continued.

Problem	Cycle time	Off-line coefficient	CV	Cost of Straight line*	Cost of U-line*
Warnecke 58-task	500	1.5	0.15	2005.36 (4)	2003.56 (4)
			0.25	2008.47 (4)	2017.67 (4)
		5	0.15	2004.28 (4)	2000.94 (4)
			0.25	2015.53 (4)	2001.40 (4)
	600	1.5	0.15	1800.41 (3)	1802.50 (3)
			0.25	1803.42 (3)	1809.10 (3)
		5	0.15	1800.31 (3)	1800.28 (3)
			0.25	1812.15 (3)	1802.68 (3)
	700	1.5	0.15	2106.24 (3)	2100.00 (3)
			0.25	2105.63 (3)	2100.61 (3)
		5	0.15	2103.09 (3)	2100.01 (3)
			0.25	2121.40 (3)	2100.55 (3)
Tonge 70-task	800	1.5	0.15	4006.04 (5)	4006.71 (5)
			0.25	4044.19 (5)	4008.08 (5)
		5	0.15	4030.83 (5)	4092.09 (5)
			0.25	4174.48 (5)	4131.10 (5)
	1000	1.5	0.15	4003.69 (4)	4004.05 (4)
			0.25	4016.54 (4)	4003.58 (4)
		5	0.15	4050.62 (4)	4096.05 (4)
			0.25	4100.00 (4)	4087.63 (4)
	1200	1.5	0.15	4801.93 (4)	4803.80 (4)
			0.25	4802.06 (4)	4802.70 (4)
		5	0.15	3885.45 (3)	3696.65 (3)
			0.25	4912.99 (4)	3772.77 (3)

*The values in parentheses indicate the number of stations.

counterparts), we sometimes observe slightly better performance of the straight line. When these solutions are carefully examined, we note that these counter-intuitive results are mainly caused by heuristic nature of the proposed beam search method. Table 2 also depicts the number of stations required in both configurations; the U-line configuration requires less number of stations in 12 instances whereas in 57 instances, both configurations yield equal number of stations. This finding is another justification of the dominance of U-line over straight-line configurations.

We also observe that the good solutions avoid incompletion on the units moving in forward direction. Because the effect of a unit moving in forward direction to incompletion cost is larger than that of a unit moving in backward direction. This negative effect is even higher if forward moving units are incomplete in earlier stations and backward moving units are incomplete in later stations. The most important design identified for the U-type configuration is that the task assignments should not make the incompletion probability of any assigned task exceed some threshold values. Thus, our findings highlight the importance of Guerriero and Miltenburg (2003) approach in solving stochastic U-lines.

Finally, a desirable good solution characteristic (or pattern) for the stochastic U-line balancing problem is the one in which the workload allocation in the upper part of the line tends to increase as we move in forward direction and workload in the lower part of the line tends to increase as we move in backward direction.

5. Conclusions

In this paper, we develop a beam search based method to solve the stochastic U-line balancing problem for the objective of minimizing total expected costs. This study, to the best of our knowledge, is the first application of the beam search methodology to the assembly line balancing problems. The proposed method is also the first heuristic procedure for the stochastic U-type problem with the total expected cost criterion. This cost consists of two components; the first one is the labour cost proportional to the number of stations in the line design and the second component is the expected incompletion cost arising from the tasks not completed within the cycle time in each station.

The proposed method is tested on the standard test problems that are originally generated for the straight-line configuration. We modify the data set by inducing variability of task times. The results indicate that the proposed method performs better than the best known heuristic (K&L) in the literature; the improvement can be as high as 24 % in straight-line configurations. Our results also reveal the fact that U-line configurations are generally better than their straight-lines in terms of the total costs for the stochastic line-balancing problem. When good solution characteristics (or patterns) of the U-line balancing problem are examined, it is observed that a good design is the one in which the workload allocation in the upper part tends to increase as we move in forward direction and the workload in the lower part tends to increase as we move in backward direction.

In this study, we also note the following important research issues: First, we handle incompletions via the policy recommended by Kottas and Lau (1973). This assumes 'hard' cycle time. But there may be other policies that can be used for this purpose. For example, the entire assembly line can wait for all the units to be completed, leading to stochastic cycle time. Another policy may be the one in which some incompleted tasks can be completed off the line whereas for the others the entire line is stopped until they are completed.

Second, it is usually assumed that workers have all the same skill levels and tasks times are worker independent. These assumptions can be relaxed by having multiple skill levels (skilled, semi-skilled and unskilled) and task times being dependent on these skills.

Third, normally distributed task times are assumed. Several other non-symmetric (skewed) distributions can be tried in future studies as well. Also, mixed model extensions of the problem should be investigated in future studies since today's production environments require multiple products assembled in the same lines. Finally, the problem can also be studied using meta-heuristics such as genetic algorithm and simulated annealing.

References

Arcus, A.L., COMSOAL—A computer method of sequencing operations for assembly lines. *Int. J. Prod. Res.*, 1966, **4**, 259–277.

Erel, E., Sabuncuoğlu, I. and Aksu, B.A., Balancing of U-Type assembly systems using simulated annealing. *Int. J. Prod. Res.*, 2001, **39**, 3003–3015.

Ghosh, S. and Gagnon, R.J., A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems. *Int. J. Prod. Res.*, 1989, **27**, 637–670.

Guerriero, F. and Miltenburg, J., The stochastic U-line balancing problem. *Naval Res. Log.*, 2003, **50**, 31–57.
Kottas, J.F. and Lau, H.S., A cost oriented approach to stochastic line balancing. *AIIE Trans.*, 1973, **5**, 164–171.
Kottas, J.F. and Lau, H.S., A total operating cost model for paced lines with stochastic task times. *AIIE Trans.*, 1976, **8**, 234–240.
Kottas, J.F. and Lau, H.S., A stochastic line balancing procedure. *Int. J. Prod. Res.*, 1981, **19**, 177–193.
Lowerre, B.T., The HARPY speech recognition system. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, 1976.
Magazine, M.J., Doerr, K.H. and Klastorin, T.D., Synchronous unpaced flow lines with worker differences and overtime cost. *Manage. Sci.*, 2000, **46**, 421–435.
Miltenburg, J. and Wijngaard, J., The U-line balancing problem. *Manage. Sci.*, 1994, **40**, 1378–1388.
Monden, Y., *Toyota Production System*, 1993 (Industrial Engineering and Management Press, Institute of Industrial Engineers: Norcross, GA).
Ow, S. P. and Morton, T.E., Filtered beam search in scheduling. *Int. J. Prod. Res.*, 1988, **26,** 35–62.
Sabuncuoğlu, I. and Bayiz, M., Analysis of reactive scheduling problems in a job shop environment. *Euro. J. Op. Res.*, 2000, **126**, 567–586.
Sabuncuoğlu, I. and Karabük, S., A beam search-based algorithm and evaluation of scheduling approaches for flexible manufacturing systems. *IIE Trans.*, 1998, **30**, 179–191.
Salveson, M.E., The assembly line balancing problem. *J. Ind. Eng.*, 1955, **6**, 18–25.
Scholl, A. and Klein, R., ULINO: Optimally balancing U-shaped JIT assembly lines. *Int. J. Prod. Res.*, 1999, **37**, 721–736.
Silverman, F.N. and Carter, J.C., A cost-based methodology for stochastic line balancing with intermittent line stoppages. *Manage. Sci.*, 1986, **32**, 455–463.
Sparling, D. and Miltenburg, J., The mixed-model U-line balancing problem. *Int. J. Prod. Res.*, 1998, **36**, 485–501.
Suresh, G. and Sahu, S., Stochastic assembly line balancing using simulated annealing. *Int. J. Prod. Res.*, 1994, **32**, 1801–1810.
Urban, T.L., Note. Optimal balancing of U-shaped assembly lines. *Management Science*, 1998, **44**, 738–741.