The Application of Single-Pass Heuristics for U-Lines

Jaydeep Balakrishnan[1]
Chun-Hung Cheng[2]*
Kin-Chuen Ho[3]
Kum Khiong Yang[4]

(published by *Journal of Manufacturing Systems*, Vol. 28, No. 1, pp. 28-40, 2009)

***Corresponding Author**
Chun-Hung CHENG
Department of Systems Engineering & Engineering Management
The Chinese University of Hong Kong
Shatin, NT
Hong Kong SAR, CHINA
Email: chcheng@se.cuhk.edu.hk

1. Operations Management Area, Haskayne School of Business, University of Calgary, Calgary, Alberta T2N 1N4, CANADA
2. Department of Systems Engineering & Engineering Management, The Chinese University of Hong Kong, Shatin, NT, Hong Kong SAR, CHINA
3. VTC School of Business & Information Systems, 20A Tsing Yi Road, Tsing Yi Island, N.T., Hong Kong SAR, CHINA
4. Lee Kong Chian School of Business, Singapore Management University, 50 Stamford Road, Singapore 178899

Acknowledgement

We would like to thank Professor John Miltenburg of McMaster University for providing data for this study. In addition, his comments and suggestions have significantly improved the quality of this manuscript. We would also express our gratitude to anonymous reviewers for their constructive comments and suggestions.

This work was substantially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. 411205).

The Application of Single-Pass Heuristics for U-Lines

Abstract

U-lines have been adopted in many manufacturing settings as part of JIT implementation. In this paper, we examine the applicability of existing straight-line heuristics for obtaining a balance on a U-line. We modify 13 single-pass heuristics and study the effectiveness of various heuristics under different problem conditions. An extensive computational study is carried out to help identify the best heuristics. In addition, we compare recent U-line procedures with a single-pass heuristic using some literature problems. Based on a single-pass heuristic, we compare the configurations of a straight- and U-line.

***Keywords*: line balancing, U-lines, Straight-lines, Just-in-Time production, heuristics**

1. Introduction

The success of JIT in many manufacturing firms has motivated many other firms to consider JIT. As a consequence of the JIT implementation, firms are replacing their traditional straight-lines with U-shaped production lines. The experiences of U-shaped lines in an JIT setting have been documented by Hall [1983], Monden [1993], Schonberger[1982], Wantuck [1989] and Voss [1987].

In this paper, we investigate the problem of designing U-shape production lines. This problem is referred to as the U-line balancing problem. The line balancing problem for straight-lines or U-lines is concerned with the grouping of tasks required for manufacturing a product into stations. In the design of a line, the list of the tasks to be done for a product, the times required performing the tasks, and the order in which the tasks are performed must be analysed. A balance refers to a feasible grouping of the tasks into stations along the line.

A rich literature exists for the straight-line balancing problem. For procedures for obtaining a balance in a straight production line, the reader may refer to Hackman [1989], Held et al. [1963], Hoffmann [1992], Johnson [1988], and Kao and Queyranne [1982]. Comprehensive surveys in the literature such as Baybars [1986], Talbot *et al*. [1986], and Ghosh and Gagnon [1989] provide a good review of current optimal and heuristic procedures for the traditional problem.

Recently, the U-line balancing problem has received some research attention. Miltenburg and Wijngaard [1994] first present a model and solution procedures for the U-line problem. They propose a dynamic programming procedure and use it to solve problem instances up to ten tasks. For large problem instances, they use a ranked positional weight heuristic.

Since their seminal work, Aase et al [2003], Ağpak et al. [2006], Baykasoğlu [2006], Baykasoğlu and Ozbakir [2007], Chiang and Urban [2006], Erel et al. [2001], Erel et al. [2005], Gökçen and Ağpak [2006], Guerriero and Miltenburg [2002], Kim et al. [2000], Kim et al. [2006], Miltenburg [1998], Miltenburg [2002], Scholl and Klein [1999], Sparling and Mitenburg [1998], Urban [1998], and Urban and Chiang [2006] investigate different versions of the problem and propose various solution algorithms.

The complexity of the problem makes it very unlikely to find an optimal solution in polynomial time. In order to make the U-line model applicable, we have to consider heuristic procedures. Talbot *et al*. [1986] review and evaluate various single-pass heuristic decision rules for the straight-line problem. In this paper, we would like to examine the extensibility of straight-line

heuristics to U-lines. Further, we would like to evaluate the performance of these heuristic decision rules under different problem characteristics.

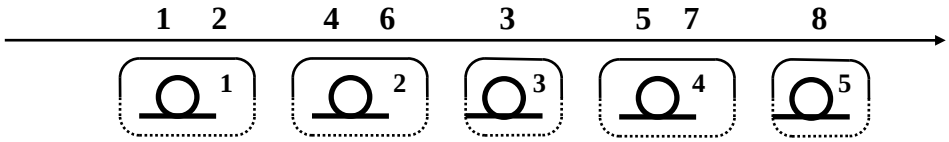
Baykasoğlu [2006] conducts a similar study assessing the performance of single-pass heuristics. Our work is different from Baykasoğlu [2006] in one major aspect. In this study, we consider walking distance in a line and travel speed of an average worker. These factors are important considerations as workers do walk from one location to another in a U-line to complete their work. Without the consideration of these factors, heuristics may generate awkward walking pattern for workers.

This paper is organised in following manner. In Section 2, a detailed description of the U-line balancing problem is given. Heuristic approaches for solving the problem are described in Section 3. We examine 13 heuristics for straight-lines and modify these heuristics for U-lines. Design of the computational study is presented in Section 4. Comparative results of the heuristics are discussed in Section 5. Section 6 provides an analysis of the computational study. We also discuss the differences and similarities between our findings and existing computational studies. Further, we compare the performance of a single-pass heuristic with recent procedures specifically designed for U-lines. In Section 7, we use a single-pass heuristic to produce and compare the configurations of a straight- and U-line. Concluding remarks are given in the last section.

2. U-line Balancing Problem

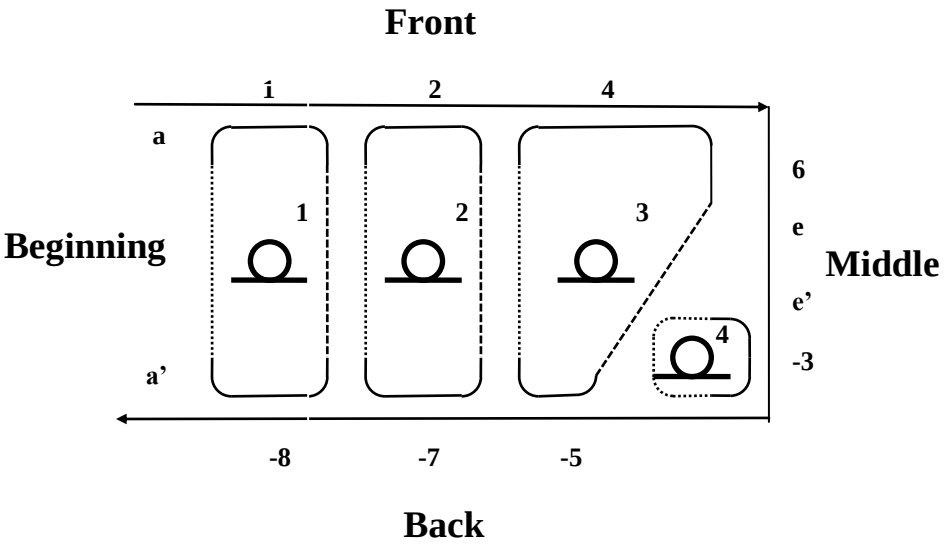
The U-line balancing problem is to assign tasks to stations arranged in a U-shaped production line. We reproduce Guerriero and Miltenburg's [2002] way to define the U-line balancing problem. Let's assume that we have a set of tasks ($J = \{1,2, ..., n\}$) with deterministic processing time (t_j), cycle time (C), and a set of precedence relations ($P = \{ (i,j) :$ *task i must precede task j* $\}$). The problem may be informally defined as: assigning all the tasks on the U-line to form a minimal number of work stations while the work content in each station should not be greater than the given cycle time.

Tasks



Stations

a. Traditional Line



b. U-Line

Key :

Operator
Path in station travelled by operation

Task distance
Return distance
Crossover distance

Figure 1. Configuration of Traditional Straight and U-line

2.1 Configuration of U-line

A straight-line only allows organising tasks sequentially in one direction to form stations. A U-line, however, permits tasks located on both side of the line to form stations. Figure 1(a) and (b) shows a traditional straight-line and a U-line, respectively.

In the U-line, *a* and *a'* are the points through which the raw material enters and the product leaves the line, respectively. The *beginning* of the U-line is *aa'* and the *middle* of the line is *ee'*. In some JIT settings, the *middle* of the line may be adjacent to another U-line. This allows an operator to walk between and work on two adjacent lines. In this manuscript, we consider a single U-line only and hence, we let $e = e'$. The positions from *a* to *e* is the *front*, and *a'* to *e'* the *back* of the line. We let $NF = || a - e ||$ be the *length of the front of the line* and $NB = || a' - e' ||$ be the *length of the back of the line*. Each task $j \in J$ requires a distance $d_j \geq 0$ on the line. Then $NF + NB \geq \sum_{j \in J} d_j$. We assume that $d_j = 1$ for each $j \in J$ in our study and thus $NF + NB = n$.

There are three types of stations on a U-line: regular, cross-over, and multi-line stations. A *regular station* has all the tasks on the same side of the line. A regular station may be formed either on the front or on the back of the line. In a *cross-over station*, tasks are located on both sides of the line while a *multi-line station* has tasks from two adjacent U-lines. Stations 1, 2 and 3 in Figure 1(b) are cross-over stations, and station 4 is regular. All stations on the traditional line are regular (e.g., Figure 1(a)). Since we only focus on a single U-line in this work, we do not have any multi-line station. The reader may refer to Miltenburg [1998] for an example of a multi-line station.

2.2 Feasible subsets and sequences

In Guerriero and Miltenburg [2002], a feasible subset *s* is composed of two subsets s^f and s^b. Subset $s^f \subseteq s$ is a set of tasks in $s \subseteq J$ where for each $j \in s^f$ all predecessors of *j* are also in s^f. Note that $| s^f | \leq NF$. Subset $s^b \subseteq s$ is a set of tasks in *s* where for each $s^b \subseteq s$ all successors of *j* are also in s^b. Also note that $| s^b | \leq NB$. A negative (-) sign is assigned to each task number in s^b. For example, $s = \{1,2,4,-7,-8\} \subset J$ mean that $s^f = \{1,2,4\}$ and $s^b = \{-7,-8\}$. Operators such as $\subset$ and $\in$ ignore the negative sign preceding a task number.

Let $\prod(s)$ denote the set of *feasible sequences* for a feasible subset $s = s^f \cup s^b \subseteq J$. A feasible sequence $\pi(s) \in \prod(s)$ is of the form $\pi(s) = j_1\_j_2\_...\_j_u\_j_v\_...\_j_l$ with a negative sign for a $j \in s^b$. A sequence $\pi(s)$ is a feasible sequence if the following two conditions are satisfied:

(1) If $j_v \in s^f$ and $(j_u, j_v) \in P$, then j_u must precede j_v in $\pi(s)$; and

(2) If $j_u \in s^b$ and $(j_u, j_v) \in P$, then j_v must precede j_u in $\pi(s)$.

A feasible sequence $\pi(s)$ for a feasible subset s specifies the positioning of the tasks on the line. The tasks are removed one at a time from left to right in $\pi(s)$ for assignment. A task will be placed in "the next available position" on the *front* of the line if it belongs to s^f. A task will be placed in "the next available position" on the *back* if it belongs to s^b. By "the next available position" we mean start at point *a* or *a'* (at *a'* if the task has a negative sign) and move away from these points until a position is available.

Suppose that we plan to assign tasks $J = \{1,2, ..., 8\}$ to the line as shown in Figure 1(b) and, we assume $NF = 4$ tasks to the front and $NB = 4$ to the back. Consider Subset $s = \{1,2,4,-7,-8\} \subseteq J$ and the sequence $\pi(s) = 1\_2\_-8\_-7\_4$. As $\pi(s)$ is read from left to right, task 1 is assigned to the next available position on the front of the line; namely the position starting at point *a*. Then task 2 is assigned to the next available position on the front of the line. As a negative sign is associated with task 8, it is assigned to the next available position on the back of the line, namely the position starting at point *a'*. Following in this way, we assign task 7 to the next position on the back of the line and task 4 to the next position on the front of the line. Now all the tasks have been assigned. Note that different feasible sequences (such as *1_-8_2_-7_4*, *1_2_4_-8_-7* and *-8_1_2_4_-7*) all have the same task assignment on the line.

2.3 Assignment of tasks to stations

A feasible sequence $\pi(s) \in \Pi(s)$ for a feasible subset s assigns the tasks to positions on the line. Tasks located close together are grouped into stations. Let $l = |s|$ be the number of tasks in s. The grouping of tasks into stations is obtained using the following procedure:

Starting at the beginning of a feasible sequence $\pi(s)$, assign as many consecutive tasks as possible to the first station without violating the given cycle time. In Guerriero and Miltenburg [2002], $\pi(s) = [j_1\_j_2...j_u]\_j_{u+1}\_j_{u+2}\_...\_j_l$ is used to indicate that tasks $A_1 = \{j_1,j_2,...,j_u\}$ are assigned to the first station. Then starting from the beginning of the remaining sequence, assign as many consecutive tasks as possible to the next work station, and so on. The resulting grouping of tasks is called the induced assignment for $\pi(s)$. Let $\pi^m(s)$ denote the sequence of tasks in $\pi(s)$ assigned to station $m = 1, 2, ...$ Then the induced assignment may be written as $\pi^1(s)\_\ \pi^2(s)\_...\_\ \pi^r(s)$ where r is the last station.

2.4 Costs

The costs of the stations are used to evaluate the performance of the assignment of the tasks on the U-line. The *work content* in a station is total time an operator at the station requires to travel to the location of the tasks and process them. The total task processing time in station m is $\sum_{j \in A_m} t_j$, where A_m is the set of the tasks assigned to station m. the total *task distance* the operator needs to travel for each production cycle at station m is $\sum_{j \in A_m} d_j$. As we mentioned earlier, $d_j = 1$ for each $j \in J$, so $\sum_{j \in J} d_j = n$. Let $d_m^{(r)}$, and $d_m^{(c)}$ denote the return, and cross-over distances, respectively, in station m. *Return distance* is the distance an operator travels when the last task in the station is completed and he/she returns to the first task. *Cross-over distance* is the distance an operator travels when the last task on the front of the line is completed and he/she crosses to the back of the line to first task to be processed there. *Cross-over distance* only occurs in stations that have tasks on the front and back of the line and thus will not appear in traditional straight-line. Notice that the traditional line is a special case of the U-line where all stations are regular stations, return distance equals total task distance, and cross-over distance is zero.

Let c denote the time required travelling one unit of distance and $\tau(\pi^m(s))$ denote the *work content* of station m. In Guerriero and Miltenburg [2002], the work content of station m is given as: $\tau(\pi^m(s)) = \sum_{j \in A_m} (t_j + c \times d_j) + c \times (d_m^{(r)} + d_m^{(c)})$. The assignment of tasks on the U-line must satisfy $\tau(\pi^m(s)) \leq C$ for each $m = 1,2...r$, where C denotes *cycle time*, the maximum work content of a station. We assume that $t_j + 2 \times c \times d_j \leq C$ for each $j \in J$. We let $T(\pi(s))$ denote the *total cost* of the assignment for all tasks, and define this cost to be the total time required to produce the product. Then we have $T(\pi(s)) = (r - 1) \times C + \sum_{j \in \pi^r(s)} (t_j + c \times d_j) + c \times (d_r^{(r)} + d_r^{(c)})$.

With any feasible subset $s \subseteq J$ may be associated with a minimum cost $X(s) = min \{ T(\pi(s)): \pi(s) \in \prod(s) \}$. In particular if $s = J$, then $X(J)$ gives the minimum number of stations r for the entire line, and the minimum time needed in the last station.

3. Heuristic Algorithms

For the traditional straight problem, many heuristics have been proposed. These decision rules vary from simple list processing procedures to optimal-seeking procedures. Talbot *et al.*

[1986] classify the heuristic rules into four categories: "Single-Pass Decision Rules" which only consider a single attribute of each assembly task to implement a prioritising scheme for task assignment, "Composite Decision Rules" which are a composite of the "Single-Pass Decision Rules", "Backtracking Decision Rules" which attempt to improve the solution using backtracking approaches, and "Optimal Seeking Decision Rules" which find the optimal solutions.

In this paper, we consider 13 single-pass, single attribute, priority dispatch scheduling rules such as Maximum Ranked Positional Weight (Helgeson and Birnie [1961]), Minimum Lower Bound (Talbot *et al*. [1984]), Minimum Upper Bound (Talbot *et al*. [1984]), etc. Based on these heuristic rules for the traditional straight-line, we modify them for the U-line. Both maximum task time and random task assignment are the same for the straight-line and U-line (see Table 1). As all heuristics for the straight-line balancing consider the assignment of the tasks in a forward direction only, we add a backward direction for task assignments so that all of them can be used for the U-line balancing.

Talbot *et al*. [1986] is the first to conduct a comprehensive study of single-pass heuristics for a straight-line. In their experiment, they ignore walking distance and travel speed. This may be fine for a straight-line as it only allows walking along the line.

Baykasoğlu [2006] conducts a similar study to assess the performance of single-pass heuristics for obtaining a balance for a U-line. In the design of his experiment, walking distance and travel speed is not considered. Although heuristics may generate solutions with excellence measures (i.e., in terms of number of workstations required, smoothness index, and line efficiency), these solutions may give awkward walking pattern for workers in the line.

Figure 2 shows the precedence diagram of Jackson's 11-element problem. When cycle time (C) is 14, MAXRPW solutions without walking distance and travel speed considerations for a straight- and U-line are given in Figure 3(a) and 3(b), respectively. In Figure 3(b), the worker at station 2 walks a long way from task 1 to task 6 and from task 8 to task 1 and the worker at station 3 also walks a long distance from task 4 to task 2 and from task 2 to task 3. This awkward pattern makes the solution difficult to implement in a line, as workers may possibly interfere with each other.

In our experiment, we try to address the limitation of the previous studies. Walking distance and travel speed are explicitly considered in our model. A MAXRPW solution with these considerations is shown in Figure 3(c). In this case, awkward walking pattern can be avoided.

Each decision rule included in this section consists of a simple, computationally efficient, list-processing procedure that assigns tasks to stations according to its forward and backward priorities. The following procedure shows how the tasks can be assigned to different stations:

1. Each task is assigned a numerical forward and backward priorities specified by the logic of the heuristic decision rule.
2. Create a new station.
3. While not all of the tasks have been assigned, do the following:
    a. Tasks that precedence, cycle time and *NF* (*NB*) feasible are placed on the available list.
    b. While the available list is not empty, do the following
        i. The task on the available list with the highest or lowest priority required by the decision rule in use is assigned to the current station. Ties are broken randomly.
        ii. Compute the costs related to the task assignment and the remaining cycle time of the current station.
        iii. Update the available list to reflect the tasks that are now precedence and NF/NB feasible. Create a new station if the current station cannot include any additional task in the available list.

Rules	Reference	Basis for Determining Forward Task Priority		Basis for Determining Backward Task Priority	
		Notation	Priority Function	Notation	Backward Priority Function
1. Maximum Ranked Positional Weight (MAXRPW)	Helgeson & Birnie [1961]	MAXRPW_F	$RPW\_F_i = t_i + \sum_{j \in S_i} t_j$	MAXRPW_B	$RPW\_B_i = t_i + \sum_{j \in P_i} t_j$
2. Maximum Total Number of Follower / Predecessors Tasks (MAXTFOL)	Talbot *et al*. [1984]	MAXTFOL	NS_i	MAXTPRE	NP_i
3. Maximum Task Time (MAXDUR)	Moodie & Young [1965]	MAXDUR	t_i	MAXDUR	t_i
4. Maximum Number of Immediate Follower / Predecessors Tasks (MAXIFOL)	Tonge [1961]	MAXIFOL	NIS_i	MAXIPRE	NIP_i
5. Minimum Slack (MINTSLK)	Talbot *et al*. [1984]	MINTSLK_F	$UB\_F_i - LB\_F_i$	MINTSLK_B	$UB\_B_i - LB\_B_i$
6. Random Task Assignment (RANDOM)	Arcus [1963]	RANDOM	random(uniform)	RANDOM	random(uniform)
7. Minimum Lower Bound (MINLB)	Talbot *et al*. [1984]	MINLB_F	$LB\_F_i = [\ RPW\_B_i / C\ ]^+$	MINLB_B	$LB\_B_i = [\ RPW\_F_i / C\ ]^+$
8. Minimum Upper Bound (MINUB)	Talbot *et al*. [1984]	MINUB_F	$UB\_F_i = n + 1 - [\ RPW\_F_i / C\ ]^+$	MINUB_B	$UB\_B_i = n + 1 - [\ RPW\_B_i / C\ ]^+$
9. Minimum Task Number (MINTSKNO)	Arcus [1963]	MINTSKNO_F	task number, i	MINTSKNO_B	task number, n - i + 1
10. Maximum Average Ranked Positional Weight (MAXAVGRPW)	Talbot *et al*. [1984]	MAXAVGRPW_F	$RPW\_F_i / (NS_i + 1)$	MAXAVGRPW_B	$RPW\_B_i / (NP_i + 1)$
11. Minimum Upper Bound Divided by the Total Number of Followers / Predecessors (MIN(UB/TFOL))	Talbot *et al*. [1984]	MIN(UB_F /TFOL)	$UB\_F_i / (NS_i + 1)$	MIN(UB_B/TPRE)	$UB\_B_i / (NP_i + 1)$
12. Maximum Task Time Divided by Task Upper Bound (MAX(DUR/UB))	Talbot *et al*. [1984]	MAX(DUR/UB_F)	t_i / UB_F_i	MAX(DUR/UB_B)	t_i / UB_B_i
13. Maximum Total Task Followers / Predecessors Divided by Task Slack (MAX(TFOL/SLK))	Talbot *et al*. [1984]	MAX(TFOL/SLK_F)	$NS_i / (UB\_F_i - LB\_F_i)$	MAX(TPRE/SLK_B)	$NP_i / (UB\_B_i - LB\_B_i)$

C: Station Cycle Time.
n: The number of tasks to be balanced into work stations.
$NS_i(NP_i)$: The total number of tasks which succeed (precede) task i (i.e. the number of elements of $S_i(P_i)$).
$NIS_i(NIP_i)$: The number of tasks which must immediately succeed (precede) task i.
$S_i(P_i)$: The set of tasks which must succeed (precede) task i.
t_i: Assembly time required to complete task i.
$[X]^+$: The smallest integer greater than or equal to X.
X_F: Forward priority function.
X_B: Backward priority function.

Table 1. Single-Pass Decision Rules

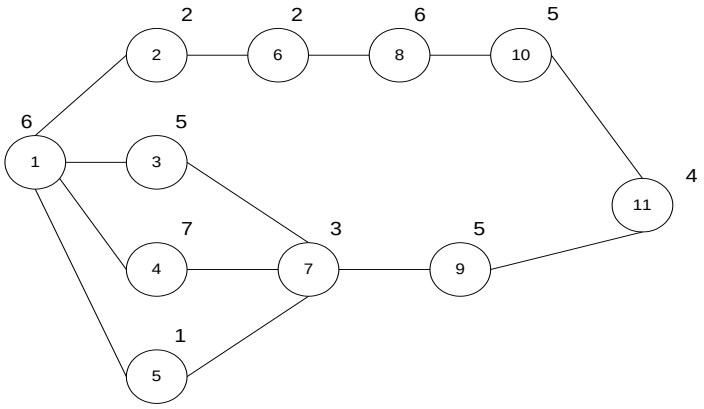
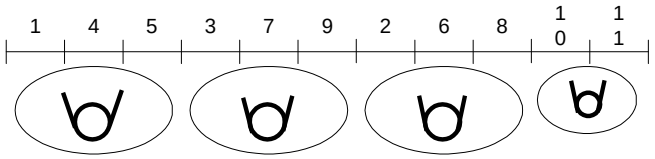
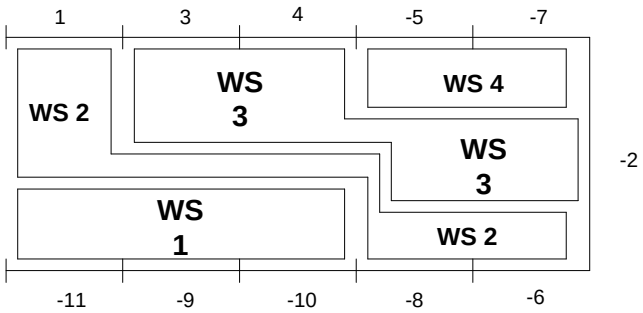


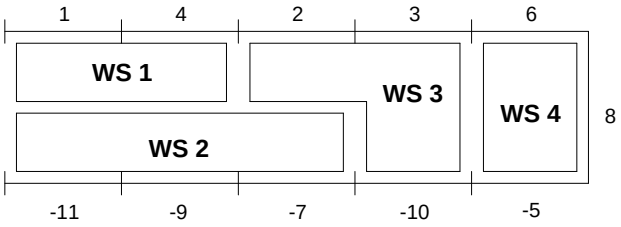
Figure 2: Precedence diagram of Jackson's 11-element Problem



(a) A straight-line solution



(b) A U-line solution without walking distance and travel speed



(c) A U-line solution with walking distance and travel speed consideraions

Figure 3: Straight- and U-line solutions with cycle time = 14

4. Design of Computational Study

The 13 heuristics are used to solve well-known line balancing problem sets in the literature. The purpose of this computational study is to find the input conditions under which heuristic(s), outperform(s) others in balancing U-lines based on different performance measures.

4.1 Test Problems

Problems are taken from Talbot *et al.* [1986] to investigate the performance of the modified heuristics. Notice that we ignore some problem instances in which the condition: $t_j + 2 \times c \times d_j \leq C$ for each $j \in J$ is violated.

4.1.1 Problem Set One

This data set is called Literature Problems in Talbot *et al.* [1986]. This set contains 12 literature problems, ranging in size from 7 to 111 tasks. Each problem will be solved for several cycle times except the Bowman [1960] problem (see Talbot and Patterson [1984]). The precedence order strength/density of the literature problems varies from 0.292 (Arcus [1963]) to 0.3056 (Jaeschke [1964]). The order strength/density of a problem is the ratio of the total number of precedence relations to the maximal number of precedence relations in the problem.

4.1.2 Problem Set Two

This data set is called "Difficult" Problem Set One in Talbot *et al.* [1986]. In this problem set, five 50-task and five 100-task assembly line networks with order strength equal to 0.2 are randomly generated. Each has eleven trial cycle times range from 10000, 11000, ... to 20000. In these problems, the processing time of each task is generated by uniform distributed randomly variable, ranging from 1 to 10,000.

4.1.3 Problem Set Three

This data set is called "Difficult" Problem Set Two in Talbot *et al.* [1986]. This set contains the same networks as in Set Two, except that all the odd processing times are increased by one unit to make them even numbers. Also all even cycle times are increased by one unit to make them odd numbers. The eleven cycle times on each problem ranges from the largest processing time to two

times of the largest processing time with a cycle time being 10% more than the previous cycle time. All cycle times are rounded up to the nearest integer.

4.1.4 Problem Set Four

This data set is called Main Experimental Data Set in Talbot *et al.* [1986]. This data set contains 120 problems: sixty 50 task and sixty 100 task assembly networks. Task processing times are derived from a binomial distribution. The density of an assembly network is D=2d/n(n+1) where d is the total number of relations in P and n is the number of tasks. We follow Talbot et al. and use D=20%, 30%, 50%, and 80%.

4.2 Input Parameters

Each problem instance is defined by a number of parameters:

1. Problem number, $u = 1,2, ...$
2. Number of tasks, n: The smallest problem instance has 7 tasks while the largest problem has over 100 tasks.
3. Cycle time, C: The time which is available in each station to perform all the tasks assigned to the station.
4. Precedence order strength (density), D: Let d denote the total number of precedence relations in P. The maximum number of precedence relations, $d_{max} = n \times (n-1)/2$. Then, $D = d/d_{max} = 2d / [n \times (n-1)]$ where $D \leq 1.0$.
5. Travel time, c: The time a worker requires to travel a unit of distance. We use two travel times for each problem instance, (i.e., $c = 0.05 \times E(t)$, and $c = 0.01 \times E(t)$), where $E(t)$ is the expected value of processing times which is defined as $E(t) = \sum_{j \in J} t_j / n$.
6. Length of the line, NF, NB. We consider the configuration of the U-line in which $NF = || a - e || = [n/2]^- + 1$, $NB = || a' - e' || = [n/2]^-$ and $NF + NB = n$, where $[X]^-$ is the largest integer that is smaller than or equal to X, and task distances are $d_j = 1$ for $j = 1,2, ..., n$,

7. Width fraction, f_w. Width fraction is the ratio of number of tasks along the width, n_w, to number of tasks. Since n_w must be an integer, we set that $n_w = [n \times f_w]^-$. Note that if n is an odd number, then n_w must be an odd number to maintain the U-line is evenly divided into two sides. Also, if n is an even number, then n_w must be an even number. In our experiments, we consider $f_w = 0.1$.

4.3 Derived Input Parameters

Based on input parameters, we find the following derived parameters:

1. Ratio of the number of tasks to a lower bound on the number of stations, n/r^*: A larger C may require fewer number of stations, and thus, make the ratio of n/r^* increase. The lower bound on the number of stations, r^*, is defined as :

   $$r^* = [(\sum_{j \in J} t_j + 2 \times c \times n) / C]^+$$

   $[X]^+$ is the smallest integer that is greater than or equal to X.

4.4 Output Variables

A number of output variables are calculated from a solution:

1. Total cost, $T(\pi(s))$ or (T in short): Total cost i.e. $T(\pi(s)) = (r - 1) \times C + \sum_{j \in \pi r(s)} (t_j + c \times d_j) + c \times (d_r^{(r)} + d_r^{(c)})$ measures the quality of a solution.
2. Ratio of the number of stations to a lower bound on the number of stations, r/r^*: This ratio is the simplest way to evaluate how well the heuristic performs. When r/r^* equals to 1, an optimal solution is found.
3. Expected idle time, $E(I)$: This measures the average idle time of an operator (station).

$E(I) = \sum_{m=1} I_m / r$, where $I_m = C - \tau(\pi^m(s))$

4. Squared co-efficient of variation of idle time, *SCV(I)*: This measures the distribution of workloads among operators.
   $SCV(I) = \sigma^2(I) / (E(I))^2$, where $\sigma^2(I) = [\ \sum_{m=1}(I_m - E(I))^2\ ] / r$
5. Expected Walking Time, $E(W_m|c)$: This measure the average time an operator spends on travelling.
   $E(W_m|c) = \sum_{m=1} W_m / r$, where W_m = walking time in station m / $\tau(\pi^m(s))$
   Note that the walking time is related to the travel time (i.e., *c*), as walking time = $c \times (d_m^{(r)} + d_m^{(c)})$.

5. Comparative Results

Each problem instance in the Problem Sets One to Four are solved twice ($c/E(t)$ = 0.05 and 0.1) by the 13 heuristics described in Section 3. All algorithms were coded in C and were run on a Sun workstation. No CPU time limit was set as the entire problem instances can be solved in a reasonable time.

5.1 Problem Set One

Most of the heuristics get similar results irrespective of the travel time, $c = E(t) * 0.05$ or 0.1 (See Table 2). In other words, we cannot distinguish the heuristics using this data set only. In some problem instances, most heuristics have solutions with $r/r^* = 1.00$. This observation implies these problem instances can easily be solved as most heuristics only employ the lower bound on the number of stations to assign the tasks on the U-line. This observation is found only in Problem Set One. Hence, we believe this problem set is relatively easy to solve when compared with other problem sets.

5.2 Problem Set Two

Table 3 shows the means of five output variables obtained by 13 heuristics in two different travel times. In accordance with the results, heuristics MAXDUR and MAX(DUR/UB) always outperform other heuristics in terms of *T*, r/r^* and *E(I)*. This means if either heuristic is used to balance the tasks on the U-line, three output variables (i.e., total cost, number of required stations

and expected idle time) are less than those required by other heuristics. However, the two heuristics always have the largest *SCV(I)* values irrespective of travel times. This result indicates that if MAXDUR or MAX(DUR/UB) is used, some stations will be busier than others. Hence, these heuristics may not be suitable if *SCV(I)* is a concern.

Table 3 indicates that heuristics MAXIFOL, RANDOM, MINLB, and MINTSKNO always get high values in terms of *T*, r/r^* and *E(I)* in both travel times with few exceptions. This result shows these four heuristics may not balance the tasks on the U-line efficiently. Surprisingly, in terms of *SCV(I)*, MAXIFOL, RANDOM and MINLB always get good solutions.

Regarding the output variable $E(W_m|c)$, most heuristics produce similar results although MINLB and MAXAVGRPW give slightly better values. When $c/E(t) = 0.05$ (where *c* is travel time), all heuristics roughly spend 9-12%'s working time on walking (i.e. $E(W_m|c) \approx 0.09$-0.12). Similarly, when $c/E(t) = 0.1$, around 16-18% of the operator's work will be spent on walking. Certainly, the higher the travel time is, the larger the total cost is. However, it seems that travel time does not significantly affect the performance of heuristics.

5.3 Problem Set Three

Basically, the observations obtained from solving Problem Set Two are further confirmed by this data set. In other words, MAXDUR and MAX(DUR/UB) always perform better than other heuristics in terms of *T*, r/r^* and *E(I)* but worse in terms of *SCV(I)*. MAXIFOL, RANDOM, MINLB, and MINTSKNO always get poor solutions in terms of *T*, r/r^* and *E(I)* in both travel times whereas MAXIFOL, RANDOM and MINLB get good results for *SCV(I)* (see Table 4). Concerning about $E(W_m|c)$, all heuristics nearly spend the same percentage of work-load on walking. Travel time does not seem to affect the performance of heuristics.

Heuristics	c / E(t)	T	r/r*	E(I)	SCV(I)	E(W)
MAXRPW	0.05	26271.0639	1.0524	105.6930	2.2620	0.0996
	0.1	30944.6450	1.0860	97.9972	2.3420	0.1799
MAXTFOL	0.05	25992.1951	1.0534	90.2310	2.0771	0.0979
	0.1	30682.9952	1.0943	137.5616	2.5846	0.1778
MAXDUR	0.05	25933.5207	1.0524	131.0104	2.2370	0.1014
	0.1	29956.9683	1.0841	124.7382	2.3801	0.1824
MAXIFOL	0.05	26820.0092	1.0889	194.9038	1.3343	0.0982
	0.1	30981.3379	1.1034	241.0877	1.3983	0.1712
MINTSLK	0.05	26359.5316	1.0710	120.5661	2.0580	0.1062
	0.1	30996.7324	1.0956	140.5389	2.0701	0.1819
RANDOM	0.05	26914.7920	1.0842	164.1719	1.4341	0.1033
	0.1	31373.3655	1.1239	209.2169	1.4842	0.1763
MINLB	0.05	26990.7491	1.1033	194.8585	1.4132	0.1023
	0.1	31787.7148	1.1257	242.0547	1.1935	0.1758
MINUB	0.05	26151.8730	1.0624	123.6645	2.0538	0.1029
	0.1	30846.1772	1.0905	133.2800	1.9716	0.1814
MINTSKNO	0.05	26613.8800	1.0642	117.6947	1.8309	0.0986
	0.1	32328.7643	1.1023	133.6683	2.0390	0.1813
MAXAVGRPW	0.05	26010.8606	1.0873	163.7673	2.0622	0.1032
	0.1	29880.2586	1.1200	175.5897	2.1389	0.1824
MIN(UB/TFOL)	0.05	25971.5217	1.0525	87.6441	2.1540	0.0977
	0.1	30641.5115	1.0919	124.6565	2.4104	0.1778
MAX(DUR/UB)	0.05	26025.7423	1.0552	146.8453	2.3974	0.1034
	0.1	30135.1233	1.0852	144.2545	2.6585	0.1837
MAX(TFOL/SLK)	0.05	26235.8312	1.0541	88.0461	2.0894	0.0991
	0.1	30616.8806	1.0916	125.7331	2.2786	0.1766

Table 2: Mean of output variables for Problem Set One in $c = E(t) * 0.05$ and 0.1

Heuristics	c / E(t)	T	r/r*	E(I)	SCV(I)	E(W)
MAXRPW	0.05	477807.1144	1.1210	1450.1231	1.4356	0.1140
	0.1	520046.4902	1.1224	1626.5130	1.3972	0.1774
MAXTFOL	0.05	480778.4788	1.1263	1494.9414	1.4173	0.1150
	0.1	523960.0420	1.1268	1697.2560	1.2096	0.1757
MAXDUR	0.05	458563.3516	1.0779	1094.9541	2.6773	0.1054
	0.1	500802.6439	1.0783	1252.9215	2.3198	0.1708
MAXIFOL	0.05	485109.6791	1.1362	1789.9749	1.1034	0.1043
	0.1	531861.6102	1.1427	1960.7025	0.9716	0.1719
MINTSLK	0.05	478951.5894	1.1221	1565.2906	1.2984	0.1078
	0.1	521801.5783	1.1216	1698.6712	1.1939	0.1731
RANDOM	0.05	492178.0281	1.1494	1899.4153	1.1093	0.1071
	0.1	543281.8671	1.1645	2222.8548	0.9282	0.1720
MINLB	0.05	486233.7172	1.1401	2018.2087	0.9717	0.0936
	0.1	534849.6935	1.1480	2146.8492	0.9465	0.1653
MINUB	0.05	478068.8647	1.1201	1499.5003	1.4163	0.1105
	0.1	521863.7554	1.1227	1683.1060	1.2270	0.1737
MINTSKNO	0.05	485352.9925	1.1348	1552.7042	1.4213	0.1187
	0.1	526548.5571	1.1313	1727.5795	1.1885	0.1775
MAXAVGRPW	0.05	471953.3591	1.1059	1527.1809	1.3855	0.0991
	0.1	518364.2513	1.1155	1718.8392	1.2399	0.1698
MIN(UB/TFOL)	0.05	482543.0897	1.1297	1511.4818	1.3486	0.1164
	0.1	523668.6121	1.1272	1654.5337	1.2375	0.1782
MAX(DUR/UB)	0.05	454311.2584	1.0684	1016.3205	3.0272	0.1038
	0.1	497555.2661	1.0706	1153.1258	2.5188	0.1714
MAX(TFOL/SLK)	0.05	480905.1206	1.1282	1515.1986	1.5606	0.1154
	0.1	522841.9265	1.1250	1633.7315	1.2845	0.1782

Table 3: Mean of output variables for Problem Set Two in $c = E(t) * 0.05$ and 0.1

Heuristics	c / E(t)	T	r/r*	E(I)	SCV(I)	E(W)
MAXRPW	0.05	477724.5072	1.1247	1445.0662	1.5532	0.1141
	0.1	517377.3179	1.1190	1600.0982	1.2953	0.1775
MAXTFOL	0.05	481943.7747	1.1315	1515.5698	1.3797	0.1143
	0.1	522669.8448	1.1297	1747.1296	1.2282	0.1759
MAXDUR	0.05	456499.6700	1.0759	1059.4125	3.0044	0.1029
	0.1	496916.6182	1.0750	1226.7416	2.5665	0.1714
MAXIFOL	0.05	484940.3078	1.1394	1760.6752	1.1278	0.1045
	0.1	529908.9691	1.1426	1994.8919	0.9847	0.1710
MINTSLK	0.05	478015.6275	1.1222	1511.7372	1.3092	0.1078
	0.1	518906.0051	1.1181	1714.4586	1.2410	0.1710
RANDOM	0.05	497427.6991	1.1665	1975.2438	1.0403	0.1112
	0.1	541340.4538	1.1674	2238.4754	0.9133	0.1750
MINLB	0.05	486034.0772	1.1418	1950.9701	1.0087	0.0954
	0.1	531308.2982	1.1456	2165.8490	0.9401	0.1653
MINUB	0.05	478209.2084	1.1246	1496.5710	1.3517	0.1106
	0.1	519375.3893	1.1231	1698.7288	1.2090	0.1753
MINTSKNO	0.05	486203.4147	1.1396	1543.6909	1.3520	0.1188
	0.1	523834.7867	1.1307	1725.4087	1.1959	0.1782
MAXAVGRPW	0.05	472962.8225	1.1132	1574.4395	1.3958	0.0985
	0.1	514413.2571	1.1091	1664.4821	1.2251	0.1700
MIN(UB/TFOL)	0.05	481365.1894	1.1315	1520.6938	1.4710	0.1141
	0.1	522810.3624	1.1306	1725.6047	1.2527	0.1786
MAX(DUR/UB)	0.05	454461.8781	1.0725	1025.3552	3.0579	0.1038
	0.1	494968.1403	1.0693	1148.0958	2.3860	0.1723
MAX(TFOL/SLK)	0.05	481095.8225	1.1308	1510.4338	1.4557	0.1144
	0.1	521648.7453	1.1264	1689.3268	1.1997	0.1771

Table 4: Mean of output variables for Problem Set Three in $c = E(t) * 0.05$ and 0.1

5.4 Problem Set Four

Basically, the results of Problem Set Four are very similar to those of Problem Sets Two and Three. They are given in Tables 5(a) through 5(e). Each table presents the results for a different output variable. Several observations may be derived. In terms of output variables, *T*, r/r^* and *E(I)* (See Tables 5(a), (b) and (c)), MAXDUR and MAX(DUR/UB) have the smallest mean values and the smallest 95% confidence intervals. MAXDUR and MAX(DUR/UB) perform significantly better than other heuristics. MAXIFOL, RANDOM, MINLB and MINTSKNO always get solutions with the worst total costs and the number of required stations while other heuristics fall in between. For output variable *E(I)*, the performance of MINTSKNO is not as bad as what it scored in Problem Sets Two and Three. The performances of RANDOM, MINLB and MAXIFOL are similar to what they have achieved in previous data sets. RANDOM is the worst heuristic in terms of *T*, r/r^* and *E(I)*.

The impacts of density, number of tasks and travel time on three output variables, *T*, r/r^* and *E(I)*, are very similar: (1)the higher the density, the larger the means and confidence intervals in both of three output variables, and (2)the longer the travel time, *c*, the larger the means and confidence intervals in *T*, r/r^* and *E(I)*. However, the more the number of tasks is, the larger the *T* is, but the smaller the r/r^* and *E(I)* are.

The performances of MINLB, RANDOM and MAXIFOL are better than other heuristics in the variation of idle times in different stations, i.e., *SCV(I)* (See Table 5(d)), whereas MAX(DUR/UB) and MAXDUR are worse than other heuristics. This is consistent with what we obtain from solving Problem Sets Two and Three. Moreover, other observations are obtained for different factors: the higher the density, the lower the means; the more the number of tasks, the larger the larger the *SCV(I)*; the longer the travel time, the smaller the means.

As far as the average walking time in each station is concerned, i.e., $E(W_m|c)$ (See Table 4(e)), most of the heuristics produce balances with almost the same percentage of work content in walking within the station. MINTSKNO has the largest mean. In general, 13-16% of the work content is used for walking on a station. As there is just a slightly difference among different heuristics, we believe that this output variable is less significant. $E(W_m|c)$ does not vary much for various densities. On average 14% of the work content is spent on walking. However, the more the number of tasks is, the smaller the value is. The longer the travel time is, the longer the walking time is spent on each station.

Factor	Level	N	Mean Grouping	95% Interval Estimate		
			Measure	(Lower Bound	,	Upper Bound)
Heuristics	MAXRPW	2319	868.4457	856.5850	,	880.3064
	MAXTFOL	2319	869.0444	857.2163	,	880.8725
	MAXDUR	2319	841.6812	830.1067	,	853.2558
	MAXIFOL	2319	884.0844	871.8560	,	896.3128
	MINTSLK	2319	867.5972	855.6946	,	879.4998
	RANDOM	2319	891.2583	878.9880	,	903.5285
	MINLB	2319	877.6396	865.5653	,	889.7138
	MINUB	2319	865.3818	853.5753	,	877.1883
	MINTSKNO	2319	878.9261	867.0204	,	890.8319
	MAXAVGRPW	2319	866.7163	854.6977	,	878.7349
	MIN(UB/TFOL)	2319	869.2535	857.4145	,	881.0926
	MAX(DUR/UB)	2319	838.7078	827.2338	,	850.1818
	MAX(TFOL/SLK)	2319	869.3462	857.5001	,	881.1924
Density	20%	7410	857.7603	851.3069	,	864.2137
	30%	7618	863.3378	856.6599	,	870.0157
	50%	7579	868.0868	861.3210	,	874.8525
	80%	7540	883.9420	877.4458	,	890.4381
Tasks	50	15041	587.2450	586.3007	,	588.1893
	100	15106	1148.1736	1146.5502	,	1149.7971
Travel Times	c = E(t) * 0.05	15600	822.5605	818.2640	,	826.8569
	C = E(t) * 0.1	14547	917.3795	912.4409	,	922.3181

Table 4(a): Analysis of mean group measures in Problem Set Four – *T*

Factor	Level	N	Mean Grouping	95% Interval Estimate		
			Measure	(Lower Bound	,	Upper Bound)
Heuristics	MAXRPW	2319	1.1890	1.1858	,	1.1923
	MAXTFOL	2319	1.1901	1.1868	,	1.1933
	MAXDUR	2319	1.1542	1.1507	,	1.1577
	MAXIFOL	2319	1.2086	1.2051	,	1.2122
	MINTSLK	2319	1.1879	1.1844	,	1.1913
	RANDOM	2319	1.2188	1.2153	,	1.2223
	MINLB	2319	1.2005	1.1969	,	1.2040
	MINUB	2319	1.1851	1.1817	,	1.1884
	MINTSKNO	2319	1.2041	1.2008	,	1.2074
	MAXAVGRPW	2319	1.1864	1.1828	,	1.1900
	MIN(UB/TFOL)	2319	1.1906	1.1873	,	1.1939
	MAX(DUR/UB)	2319	1.1497	1.1463	,	1.1531
	MAX(TFOL/SLK)	2319	1.1905	1.1872	,	1.1938
Density	20%	7410	1.1743	1.1724	,	1.1762
	30%	7618	1.1752	1.1733	,	1.1770
	50%	7579	1.1898	1.1880	,	1.1917
	80%	7540	1.2160	1.2141	,	1.2180
Tasks	50	15041	1.2030	1.2015	,	1.2044
	100	15106	1.1748	1.1736	,	1.1761
Travel Times	c = E(t) * 0.05	15600	1.1760	1.1748	,	1.1771
	C = E(t) * 0.1	14547	1.2027	1.2012	,	1.2042

Table 4(b): Analysis of mean group measures in Problem Set Four – r/r^*

Factor	Level	N	Mean Grouping	95% Interval Estimate		
			Measure	(Lower Bound	,	Upper Bound)
Heuristics	MAXRPW	2319	3.2282	3.1955	,	3.2609
	MAXTFOL	2319	3.2503	3.2179	,	3.2828
	MAXDUR	2319	2.9794	2.9426	,	3.0161
	MAXIFOL	2319	3.7948	3.7608	,	3.8288
	MINTSLK	2319	3.3995	3.3657	,	3.4332
	RANDOM	2319	3.9087	3.8761	,	3.9413
	MINLB	2319	3.7085	3.6770	,	3.7399
	MINUB	2319	3.2822	3.2493	,	3.3151
	MINTSKNO	2319	3.2905	3.2576	,	3.3233
	MAXAVGRPW	2319	3.5337	3.4996	,	3.5677
	MIN(UB/TFOL)	2319	3.2496	3.2167	,	3.2826
	MAX(DUR/UB)	2319	2.8454	2.8086	,	2.8822
	MAX(TFOL/SLK)	2319	3.2573	3.2244	,	3.2901
Density	20%	7410	3.1376	3.1184	,	3.1568
	30%	7618	3.1920	3.1732	,	3.2108
	50%	7579	3.3924	3.3733	,	3.4115
	80%	7540	3.7305	3.7113	,	3.7497
Tasks	50	15041	3.5147	3.5001	,	3.5292
	100	15106	3.2134	3.2004	,	3.2264
Travel Times	c = E(t) * 0.05	15600	3.0329	3.0213	,	3.0444
	C = E(t) * 0.1	14547	3.7184	3.7042	,	3.7326

Table 4(c): Analysis of mean group measures in Problem Set Four – *E(I)*

Factor	Level	N	Mean Grouping	95% Interval Estimate		
			Measure	(Lower Bound	,	Upper Bound)
Heuristics	MAXRPW	2319	0.7426	0.7268	,	0.7584
	MAXTFOL	2319	0.7358	0.7204	,	0.7512
	MAXDUR	2319	1.0397	1.0146	,	1.0647
	MAXIFOL	2319	0.6370	0.6251	,	0.6490
	MINTSLK	2319	0.7214	0.7074	,	0.7353
	RANDOM	2319	0.6204	0.6092	,	0.6317
	MINLB	2319	0.6182	0.6073	,	0.6291
	MINUB	2319	0.7248	0.7103	,	0.7393
	MINTSKNO	2319	0.7082	0.6934	,	0.7230
	MAXAVGRPW	2319	0.7375	0.7217	,	0.7534
	MIN(UB/TFOL)	2319	0.7456	0.7297	,	0.7615
	MAX(DUR/UB)	2319	1.0781	1.0511	,	1.1052
	MAX(TFOL/SLK)	2319	0.7296	0.7149	,	0.7443
Density	20%	7410	0.8475	0.8360	,	0.8590
	30%	7618	0.8104	0.8004	,	0.8205
	50%	7579	0.7344	0.7252	,	0.7435
	80%	7540	0.6362	0.6293	,	0.6432
Tasks	50	15041	0.7409	0.7338	,	0.7480
	100	15106	0.7727	0.7661	,	0.7793
Travel Times	c = E(t) * 0.05	15600	0.8382	0.8310	,	0.8454
	C = E(t) * 0.1	14547	0.6696	0.6634	,	0.6757

Table 4(d): Analysis of mean group measures in Problem Set Four – *SCV(I)*

Factor	Level	N	Mean Grouping	95% Interval Estimate		
			Measure	(Lower Bound	,	Upper Bound)
Heuristics	MAXRPW	2319	0.1446	0.1431	,	0.1462
	MAXTFOL	2319	0.1443	0.1427	,	0.1458
	MAXDUR	2319	0.1316	0.1301	,	0.1330
	MAXIFOL	2319	0.1333	0.1319	,	0.1348
	MINTSLK	2319	0.1367	0.1352	,	0.1381
	RANDOM	2319	0.1353	0.1339	,	0.1368
	MINLB	2319	0.1317	0.1303	,	0.1332
	MINUB	2319	0.1397	0.1382	,	0.1411
	MINTSKNO	2319	0.1518	0.1500	,	0.1536
	MAXAVGRPW	2319	0.1304	0.1289	,	0.1319
	MIN(UB/TFOL)	2319	0.1447	0.1431	,	0.1462
	MAX(DUR/UB)	2319	0.1345	0.1331	,	0.1360
	MAX(TFOL/SLK)	2319	0.1442	0.1426	,	0.1457
Density	20%	7410	0.1377	0.1369	,	0.1386
	30%	7618	0.1398	0.1389	,	0.1406
	50%	7579	0.1394	0.1386	,	0.1403
	80%	7540	0.1377	0.1369	,	0.1386
Tasks	50	15041	0.1416	0.1410	,	0.1422
	100	15106	0.1358	0.1352	,	0.1363
Travel Times	c = E(t) * 0.05	15600	0.1079	0.1075	,	0.1082
	C = E(t) * 0.1	14547	0.1717	0.1714	,	0.1720

Table 4(e): Analysis of mean group measures in Problem Set Four – $E(W_m|c)$

6. Analysis of Findings

In this section, we analyse our computational results from the previous section. We first discuss the performance of the 13 heuristics. Subsequently we compare our findings with existing similar studies. Finally, we compare the performance of these heuristics with recent heuristics developed specifically for U-lines.

6.1 Performance of 13 Heuristics

In order to get a general picture of the performance of 13 single-pass heuristics, they are divided into three groups based on their performance: ***excellent***, ***fair*** and ***poor***. The ***Excellent*** group includes MAXDUR and MAX(DUR/UB); the ***poor*** group includes MAXIFOL, RANDOM, MINLB, MINTSKNO; and the ***fair*** group includes all remaining heuristics including MAXRPW, MAXTFOL, MINTSLK, MINUB, MAXAVGRPW, MIN(UB/TFOL) and MAX(TFOL/SLK). This classification is based on the ratio of r/r^*. If r/r^* equals to 1, this means the solution is optimal in terms of the number of required stations.

Both MAXDUR and MAX(DUR/UB) always employ the lowest number of stations to assign the jobs on the line compared with other heuristics. We group them together in the ***excellent*** group. However, MAXDUR or MAX(DUR/UB) always has the highest *SCV(I)*. This means that if MAXDUR or MAX(DUR/UB) is employed, some stations will have larger idle times than others. This phenomenon happens because both heuristics gives higher priorities to task with longer processing times. In stations formed in the early stage of processing, very few tasks are added to the available list and are chosen for forming stations. At the later stage of processing, tasks with smaller processing list are added and chosen for forming stations. As a result, stations formed at the later stage have lower idle times than stations at the earlier stage.

As the computation for MAXTFOL, MAXAVGRPW, MIN(UB/TFOL) and MAX(TFOL/SLK) is based on the number of successors for forward priorities (or predecessors for backward priorities), it is not surprising that they are grouped together in the same group, (i.e., the ***fair*** group). MAX(TFOL/SLK) and MINTSLK employ slack to compute priorities. MAXAVGRPW uses MAXRPW as a basis to compute forward and backward priorities. It is natural that MINTSLK and MAXRPW may be placed in the same group as MAX(TFOL/SLK) and MAXAVGRPW. In addition, MINUB achieves the "average" performance in terms of the number of stations. Hence, this heuristic belongs to the ***fair*** group.

MAXIFOL, RANDOM, MINLB and MINTSKNO are grouped together. In this group, some heuristics use ad-hoc logic. For instance, RANDOM uses uniformly distributed random numbers and MINTSKNO employs task numbers to determine forward and backward priorities. On the first glance, it appears that MAXIFOL and MAXTFOL should produce similar solutions as they use similar logic. However, MAXIFOL only takes the number of immediate followers (predecessors) into account, while MAXTFOL counts all number of followers (predecessors). As a result, MAXIFOL may have more tasks with the same priority than MAXTFOL. In this case, ties are broken randomly. Thus, this tie-breaking treatment explains why MAXIFOL and MAXTFOL do not give the similar performance.

To further understand the performance of these 13 heuristics, we examine all computational results and identify the number of non-dominated solutions produced by all heuristics. Their performance in finding non-dominated solutions to the test problems is shown in Figure 4. In terms of objective functions *r* (i.e. the number of stations) and *SCV (I)* (i.e. a measure for distribution of workload among workstations), MAXAVGRPW from the ***Fair*** group outperforms all other heuristics by a big margin. The performance of all other heuristics in the ***Fair*** group is about the same. In the ***Excellent*** group, MAX(DUR/UB) performs better than MAXDUR. MAXIFOL from the ***Poor*** group delivers a surprising result and it almost matches MAX(DUR/UB) in producing the number of non-dominated solutions.

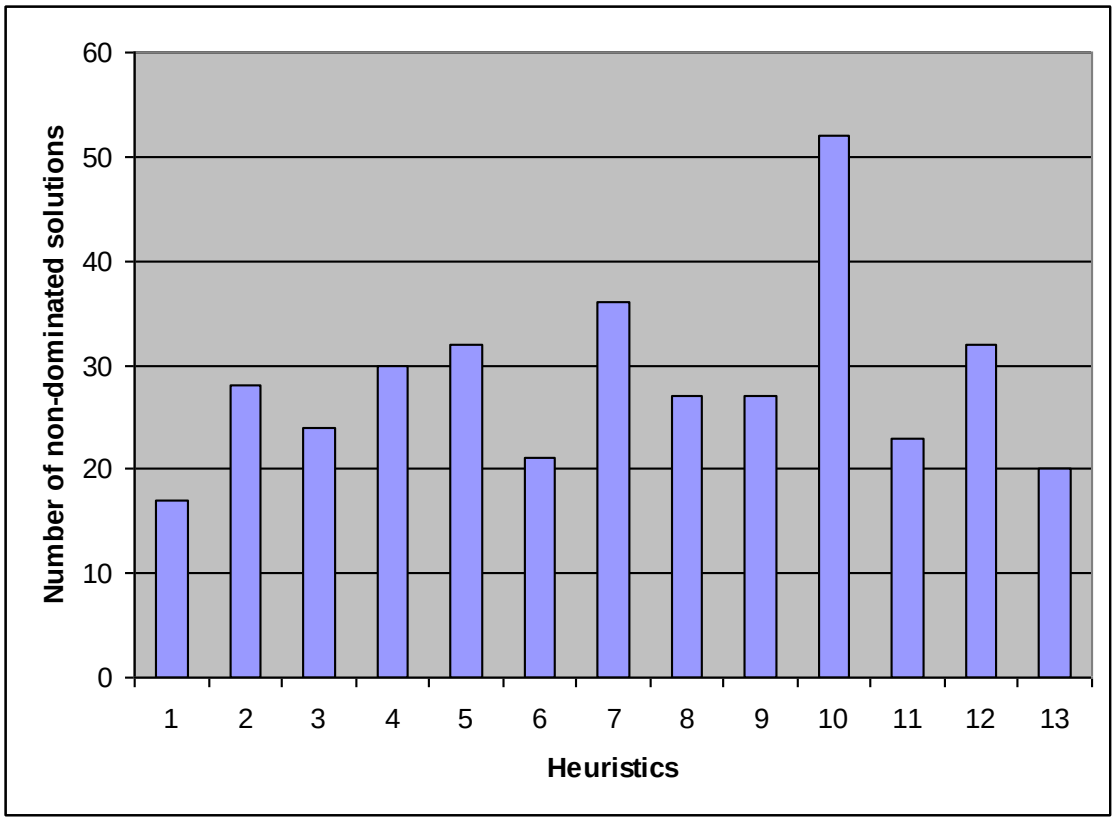


Figure 4. Number of times a heuristic found a best solution for both objectives at the same time

6.2 Comparison with Similar Studies

Our study is different from previous studies in the treatment of walking distance and travel speed. Without these considerations, a solution may produce awkward walking pattern. This problem may not be serious in a straight-line but it is noticeable in a U-line. Therefore, obvious observations in the comparison may not be valid as the underlying assumptions are different.

According to the Talbot *et al*., MAX(DUR/UB), MINUB and MAXDUR always get better results than the other heuristics whereas RANDOM and MINTSKNO always obtain poor results. In our experiment, MINUB is not as good as MAXDUR and MAX(DUR/UB) for U-lines but only provides average results. This might not be surprising as the U-line balancing problem is a generalization of the straight-line balancing problem. Heuristics for solving the straight-line balancing problem efficiently may not imply that it can solve the U-line balancing problem in the same efficient manner. Most importantly, in Problem Set Four, when the output variables T, r/r^* are considered, MINUB is the third-best, following MAX(DUR/UB) and MAXDUR. This means that MINUB provides a good balance on U-line in terms of T, and r/r^*. Like RANDOM and MINTSKNO, MAXIFOL and MINLB cannot balance the tasks on a U-line well. However, these two heuristics better perform in balancing a straight-line. This kind of difference might be explained by the same reason we have just provided.

Our finding is slightly different from Baykasoğlu's analysis. In terms of the number of workstations required, MAXRPW and MINTSLK performs the best. In our case, their performance puts them in the **fair** group. MAXDUR is ranked the second best in Baykasoğlu's study but it belongs to the **excellent** group in our finding. In the assessment of the performance of other heuristics, Baykasoğlu's results and our work are consistent.

In terms of workload distribution among workstations, Baykasoğlu adopts smoothness index (*SI*) while we uses Squared co-efficient of variation of idle time, *SCV(I)*. Although different measures are used, we still find consistent observations. MAXDUR and MAX(DUR/UB) produce unbalanced workload among workstations. MAXRPW and MINTSLK produce far better workload distributions.

6.3 Performance against Recent Heuristics

To further understand the effectiveness of the same single-pass heuristic used in the previous section, we compare it with recent procedures using literature problems specifically designed for U-lines. To make it consistent with existing heuristics, we ignore walking distance and travel speed in the calculation. The objective is to minimize the number of stations required.

The solution quality of these procedures is given in Table 5. In the table, we show the optimal number of stations needed, lower bound, and solution value. These heuristics are considered: Crossover Station Based Method (CSBM) by Ağpak [2006], U-Line Optimizer (ULINO) by Scholl and Klein [1999], and Simulated Annealing based Method (SABM) by Baykasoğlu [2006].

The computational results of MAXAVGRPW are shown in the last column. In a problem instance where MAXAVGRPW cannot produce the optimal value, its solution value is shown in italic. Out of 46, MAXAVGRPW is able to solve 39 problem instances optimally. Non-optimal solutions are found in three problems: Mitchell (28 tasks), Heskiaoff (28 tasks), and Kilbridge & Wester (45 tasks).

Single-pass heuristics were developed more than 20 years ago for obtaining a balance a straight-line. These are very simple assignment rules. Clearly, they cannot perform better than recent heuristics developed specifically for balancing a U-line. However, the strength of these heuristics is that they require very small computational effort to produce a reasonable solution to a large problem. Hence, they can be used to generate immediate solutions for other metaheuristics (such as simulated annealing, genetic search, and tabu search, etc.) to conduct further improvement.

Problem	No. of Tasks	Cycle Time	Lower Bound[1]	Optimal Value[1]	ULINO[1]	CSBM[1]	SABM[2]	MAXAVGRPW
Merten	7	6	5	6	6	6	6	6
		7	5	5	5	5	5	5
		8	4	5	5	5	5	5
		10	3	3	3	3	3	3
		15	2	2	2	2	2	2
		18	2	2	2	2	2	2
Jaeschke	9	6	7	8	8	8	8	8
		7	6	7	7	7	7	7
		8	5	6	6	6	6	6
		18	3	3	3	3	3	3
Jackson	11	9	6	6	6	6	6	6
		10	5	5	5	5	5	5
		13	4	4	4	4	4	4
		14	4	4	4	4	4	4
		21	3	3	3	3	3	3
Mitchell	28	14	8	8	8	8	8	8
		15	7	8	8	8	8	8

		21	5	5	5	5	5	*6*
Heskiaoff	28	138	8	8	8	8	8	8
		205	5	5	5	5	5	*6*
		216	5	5	5	5	5	5
		256	4	4	4	4	4	*5*
		324	4	4	4	4	4	4
		342	3	3	3	3	3	3
Kilbridge	45	57	10	10	10	10	10	10
And		79	7	7	7	7	7	*8*
Wester		92	6	6	6	6	6	*7*
		110	6	6	6	6	6	6
		138	4	4	4	4	4	*5*
		184	3	3	3	3	3	*4*
Tonge	70	176	20	21	21	21	21	21
		364	10	10	10	10	10	10
		410	9	9	9	9	9	9
		468	8	8	8	8	8	8
		527	7	7	7	7	7	7
Arcus	83	5048	15	16	16	16	16	16
		6842	12	12	12	12	12	12
		7571	10	11	11	11	11	11
		8412	9	10	10	10	10	10
		8898	9	9	9	9	9	9
Arcus	111	5755	27	27	27	27	27	27
		8847	17	18	18	18	18	18
		10027	15	16	16	16	16	16
		10743	14	15	15	15	15	15
		11378	14	14	14	14	14	14
		17067	9	9	9	9	9	9

1. Ağpak et al. [2006]
2. Baykasoğlu [2006]

Table 5: MAXRPW versus Recent Heuristics

7. Configuration of Straight- and U-Lines

Cheng et al. [2000] compare and contrast the performance of a straight- and U-line in term of product quality. They use measures in quality planning, quality control, and quality improvement to show that a U-line outperforms a straight-line. In this section, we attempt to use a literature problem to show the difference between a straight- and U-line. The literature problem we pick is Kilbridge & Wester's 45-task problem (see Talbot et al. [1986] for details of this problem). For a solution procedure, we pick the average ranked positional weight solution heuristic (MAXAVGRPW).

According to our computational study presented in the previous section, the performance of MAXAVGRPW is only fair. We do not use heuristics from the excellent group because these heuristics typically provide high variation in station workload. Figure 5 shows the straight- and U-line solutions by MAXAVGRPW with cycle time $C = 141$ and travel time $c = 0.01 \times E(t)$. Recall that $E(t)$ is the expected value of processing times which is defined as $E(t) = \sum_{j \in J} t_j / n$.

The straight- and U-line solutions use the same number of stations. However, the composition of tasks performed by a station is drastically different. In a straight-line, the length of the front of the line is n and crossover stations are not possible. Therefore, a straight-line is a special case of the associated U-line. Crossover stations provide a U-line these advantages:

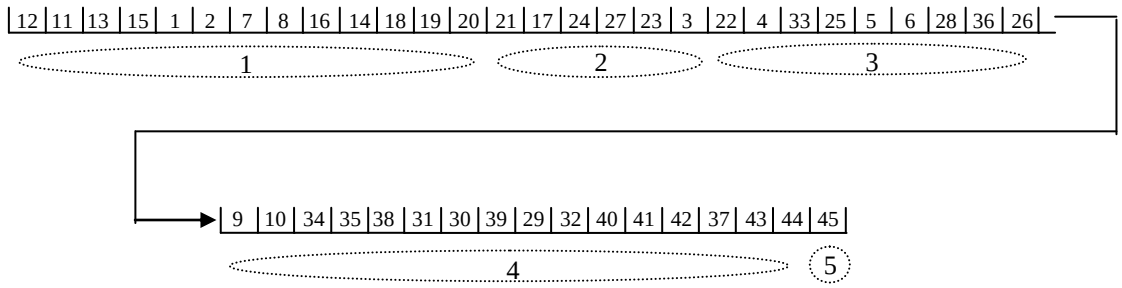
1. Crossover stations allow operators to work on both sides of a U-line. This increases the number of ways to assign tasks to stations. Hence, a straight-line cannot provide a solution better than a U-line in terms of total cost (i.e., $T(\pi(s))$). The total costs for the straight- and U-line are 569.25 and 568.25, respectively.
2. The flexibility of crossover stations also allows MAXAVGRPW to take advantage of its ability to even out the workload among stations. It is not surprising to see a better workload distribution in a U-line. In this example, $SCV(I)$ is much lower in a U-line (i.e., for the straight-line, it is 3.90 and for the U-line, it is 3.50).

On the other hand, a regular station in a straight-line only requires its operator to travel the task distance and the return distance. It is obvious that an operator in a U-line may travel more in the line. In this example problem, the total distances travelled by all 5 operators in a straight- and a

U-line are 65 units and 82 units, respectively. On average, an operator in a U-line may spend slightly more time on walking within the line. In both cases, we see that the last stations perform only one task. They stay idle most of the time. There are two ways to deal with the problem. One simple way may be increasing cycle time to eliminate the last station. The task will be performed by one of the four stations.

Another way to deal with the problem is to put two lines next to each one. In a U-line situation, we may connect this U-line and another adjacent line. The operator at the last station may move around two lines. We call such a station a multi-line station (see Miltenburg [1998]). Similarly for a straight-line situation, we allow the operator at the last station to work on two adjacent lines.

(a) Straight-line



(b) U-line

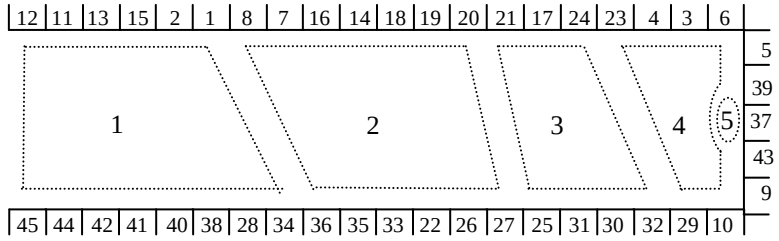


Figure 5. Straight- and U-line Solution to Kilbridge & Wester (45 tasks)

8. Concluding Remarks

In the literature, some 13 “single-pass” heuristics have been proposed for solving the line-balancing problem for a straight assembly line. In this paper, we study the line-balancing problem for a U-line and modify these 13 “single-pass” straight-line heuristics for the U-line problem. In terms of the computational experiment, over 2800 problem instances from the literature have been solved using these heuristics and the statistical results are discussed. Based on our findings, we classify 13 heuristics into three groups: *excellent*, *fair* and *poor*. Heuristics in the excellent group perform well on total cost but poorly on workload balancing. Heuristics in the fair group perform slightly worse on total cost but better on workload balancing. Further, we demonstrate the use of a heuristic in the fair group. Based on the chosen heuristic, we produce and compare the configurations of a straight- and U-line. We also discuss the benefits of crossover stations in a U-line. A comparison with recent heuristics is also reported.

References

Aase G.R., Schniederjans M.J., and Olson J.R., "U-OPT: an analysis of exact U-shaped line balancing procedures", *International Journal of Production Research*, 41, 17, 4185-4210, 2003.

Ağpak K., Çetinyokuş T., and Gökçen H., "Crossover station based simple U-type assembly line balancing", Proceedings of 5th International Symposium on Intelligent Manufacturing Systems, 118-131, May, 2006.

Baybars I., "A Survey of Exact Algorithms for the Simple Assembly Line Balancing Problem," *Management Science*, 32, 909-932, 1986.

Baykasoğlu A., "Multi-rule multi-objective simulated annealing algorithm for straight and U type assembly line balancing problems", *Journal of Intelligent Manufacturing*, 17, 217-232, 2006.

Baykasoğlu A., and Özbakır L., "Stochastic U-line balancing using genetic algorithms", *International Journal of Advanced Manufacturing Technology*, 32, 1-2, 139-147, 2007.

Cheng C.H., Miltenburg J., and Motwani J., "The effect of straight- and U-shaped lines on quality", *IEEE Transactions on Engineering Management*, 47, 3, 321-334, 2000.

Chiang W.C., and Urban T.L., "The stochastic U-line balancing problem: a heuristic procedure", *European Journal of Operational Research*, 175, 3, 1767-1781, 2006.

Erel E., Sabuncuoglu I, and Aksu B.A., "Balancing of U-type assembly systems using simulated annealing", *International Journal of Production Research*, 39, 13, 3003-3015, 2001.

Erel E., Sabuncuoglu I., and Sekerci H., "Stochastic assembly line balancing using beam search", *International Journal of Production Research*, 43, 7, 1411-1426, 2005.

Ghosh, S. and Gagnon R.J., "A Comparative Literature Review and Analysis of the Design, Balancing and Scheduling of Assembly Systems," *International Journal of Production Research*, 27, 4, 637-670, 1989.

Gökçen H., and Agpak K., "A goal programming approach to simple U-line balancing problem", *European Journal of Operational Research*, 171, 2, 577-585, 2006.

Guerriero F., and Miltenburg J., "The stochastic U-line balancing problem", *Naval Research Logistics*, 50, 1, 31-57, 2002.

Hackman S.T., Magazine M.J., and Wee T.S., "Fast, Effective Algorithms For Simple Assembly Line Balancing Problems," *Operations Research*, 37, 6, 916-924, 1989.

Hall R. W., *Zero Inventories*, Dow Jones-Irwin, Homewood, Illinois, 1983.

Held M., Karp R. M., and Shareshian R., "Assembly-line balancing -- dynamic programming with precedence constraints," *Operations Research*, 11, 442-459, 1963.

Hoffmann, T.R., "Eureka: a hybrid system for assembly line balancing," *Management Science*, 8, 1, 39-47, 1992.

Johnson R.V., "Optimally balancing large assembly lines with 'Fable' ", *Management Science*, 34, 1, 240-253, 1988.

Kao E.P., and Queyranne M., "On dynamic programming for assembly line balancing", *Operations Research*, 30, 2, 375-390, 1982.

Kim Y.K., Kim S.J., and Kim J.Y., "Balancing and sequencing mixed-model U-lines with a co-evolutionary algorithm", *Production Planning & Control*, 11, 8, 754-764, 2000.

Kim Y.K., Kim J.Y., and Kim Y.H., "An endossymbiotic evolutionary algorithm for the integration of balancing and sequencing in mixed-model U-lines", *European Journal of Operational Research*, 168, 3, 838-852, 2006.

Miltenburg G.J., "Balancing U-lines in a multiple U-line facility", *European Journal of Operational Reserch*, 109, 1, 1-23, 1998.
Miltenburg G.J., "Balancing and scheduling mixed-model U-shaped production lines", *International Journal of Flexible Manufacturing Systems*, 14, 2, 119-151, 2002.

Miltenburg G. J. and Wijngaard J., "The U-Line Balancing Problem," *Management Science*, 40, 10, 1378-1388, 1994.

Gutjahr A.L., and Nemhauser G.L., "An algorithm for the line balancing problem," *Management Science*, 11, 2, 308-315, 1964.

Monden Y., *Toyota Production System*, Second Edition, Industrial Engineering Press, Institute of Industrial Engineers, Norcross, GA, 1993.

Schonberger R.J., *Japanese Manufacturing Techniques: Nine Hidden Lessons in Simplicity*, Free Press, New York, 1982.

Sparling D., and Miltenburg J., "The mixed-model U-line balancing problem", *International Journal of Production Research*, 36, 10, 485-501, 1998.

Scholl A., and Klein R., "ULINO: optimally balancing U-shaped JIT assembly lines", *International Journal of Production Research*, 37, 4, 721-736, 1999.

Steiner G., "On the complexity of dynamic programming for sequencing problems with precedence constraints," *Annals of Operations Research*, 26, 103-123, 1990.

Talbot F.B. and Patterson J.H., "An integer programming algorithm with network cuts for solving the assembly line balancing problem", *Management Science*, 30, 1, 85-99, 1984.

Talbot F.B., Patterson J.H., and Gehrlein W.V., "A comparative evaluation of heuristic line balancing techniques," *Management Science*, 32, 4, 430-454, 1986.

Urban T.L., "Optimal balancing of U-shaped assembly lines", *Management Science*, 44, 5, 738-741, 1998.

Urban T.L., and Chiang W.C., "An optimal piecewise-linear program for the U-line balancing problem with stochastic task times", European Journal of Operational Research, 168, 3, 771-782, 2006.

Voss, C. A., *Just-In-Time Manufacture*, IFS(Publications)Ltd, U.K., 1987.

Wantuck K.A., *Just In Time for America*, The Forum Ltd., Milwaukee, WI, 1989.