HUP: Heuristic for U-Shaped Parallel Lines

Louis Plebani
Lehigh University

Sihua Chen
Binney & Smith, Inc.

Report No. 03T-008

HUP: Heuristic for U-Shaped Parallel Lines

Louis Plebani[1] and Sihua Chen[2]

[1]Department of Industrial and Systems Engineering, Harold S. Mohler Laboratory, 200 West Packer Avenue, Bethlehem, PA 18015-1582

[2] Manufacturing Development Manager, Binney & Smith, Inc., 1100 Church Lane, P.O. Box 431, Easton, Pennsylvania. 18044-0431

Abstract

Largely due to pressures of just-in-time (JIT) manufacturing, many assembly lines are designed as U-shaped assembly lines. While the straight assembly line has been studied for nearly 5 decades, the first published work on U-shaped lines was not until 1994. This paper introduces a model of a new U-shaped assembly line balancing problem, the U-shaped assembly line balancing problem where parallel stations are permitted. The model incorporates the practical constraint that parallel stations are allowed only to accommodate a task with processing time greater than the cycle time. A heuristic to solve this problem and computational results are presented.

1 Introduction

In the simple assembly line balancing problem Type I, we are given a finite set of tasks having fixed processing time $T = \{t_1, t_2, \ldots, t_N\}$, and a set of precedence relations $P = \{(x, y)\}$ specifying task x must be completed before task y. The goal is to assign the tasks to an ordered sequence of workstations such that the precedence relations are satisfied, the sum of the processing times at each station does not exceed a station uniform cycle time, and the number of stations in minimized. The straight line assembly line balancing problem was introduced by Salveson in 1955 [9]. In order to obtain increases in line efficiency, flexibility in assigning tasks into stations, and higher production rates, Freeman and Jucker suggested straight line balancing with parallel workstations in 1967 [3]. Buxey studied the practical aspects of parallel stations

including costs of duplicated equipment and difficulties of layout and transportation and concluded that the numbers of parallel stations should be tightly constrained [2]. His arguments centered around the fact that parallel workstations should not detract from the essential benefits of just-in-time flow line production. They should only be used to fit longer elements into demanded cycle time. Largely due to pressures of JIT, Miltenburg and Wijngaard introduced U-shaped line balancing in 1994 [8]. U-lines have the potential for better balancing, improved visibility and communications, fewer work stations, more flexibility for adjustment, minimization of operation travel, and easier material handling when compared to straight lines. This paper introduces a new U-shaped assembly line balancing problem, the U-shaped assembly line balancing problem where parallel stations are permitted.

The ALBP is known to be NP-Hard [7]. Solution approaches have included various heuristic approaches and exact methods using Integer Programming, Dynamic Programming and Branch and Bound. [1, 4, 10]. While exact procedures, particularly branch and bound, have had some success with small to modest size problems, heuristic procedures are needed for feasible solutions and bounds for exact methods and for use in solving variations of the ALBP using search procedures where the Type I problem is repetitively solved. We present a heuristic (HUP) for solving the U-shaped assembly line balancing problem where parallel stations are permitted.

2 U-shaped Assembly Line with Parallel Stations

A U-shaped assembly line with parallel stations can be viewed as a U-shaped line where stations are replaced with stages where a stage is two or more identical workstations operating in parallel. The capacity of the k^{th} stage is $q_k C$, where q_k is the number of identical workstations and C is the cycle time. The goal is to minimize the total idle time in the system

$$\sum_{k=1}^{M} q_k C - \sum_{i=1}^{N} t_i \tag{1}$$

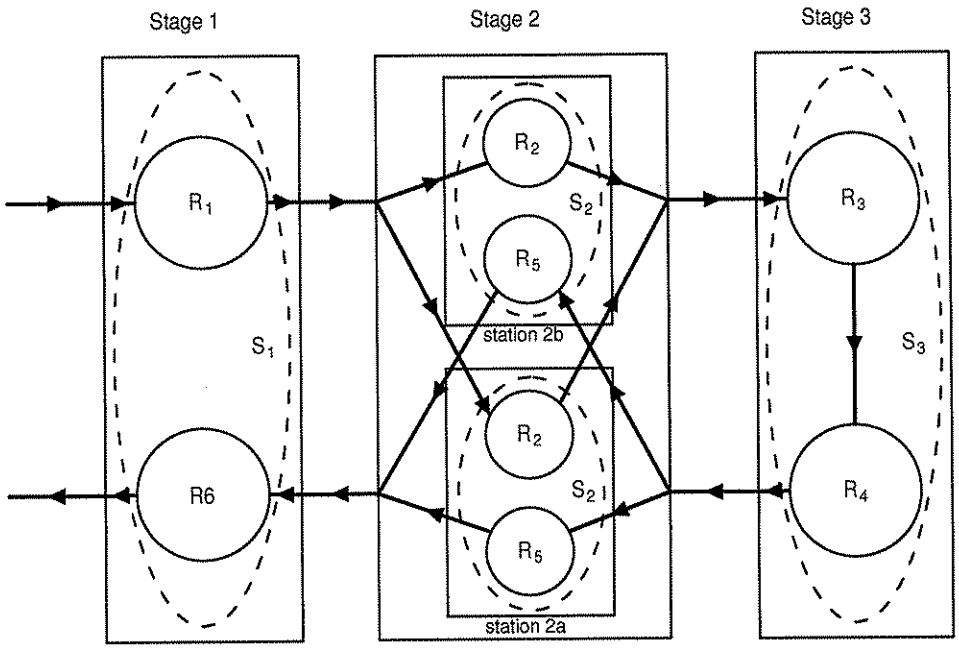

Figure 1: U-shaped line with a parallel station showing task sets and work flow

which is equivalent to minimizing $\sum q_k$. Define the following sets:

R_k, $k = 1, 2, ..M$ are the set of tasks that are processed at the k^{th} stage during the first half of the U-line,

R_{M+k}, $k = 1, 2, ..M$ as the set of tasks that are processed at the k^{th} stage during the second half of the U-line.

These sets are identified in the schematic of a 3 stage U-shaped line, with 2 parallel stations in the second stage, shown in Figure 1. A model of the U-shaped assembly line with parallel stations line balancing problem is:

$$\text{Minimize} \quad \sum_{k=1}^{M} q_k \tag{2}$$

Subject to:

$$S_k = R_k \cup R_{M+k} \tag{3}$$

$$\bigcup_{k=1}^{M} S_k = T \tag{4}$$

$$S_i \cap S_j = \emptyset, \quad \text{for all } i \neq j \tag{5}$$

$$\sum_{i \in S_k} t_i \leq q_k C, \quad k = 1, 2, .., M \tag{6}$$

$$\text{for all } (x, y) \in P : \text{if } x \in R_i \text{ and } y \in R_j, \text{ then } i \leq j \tag{7}$$

$$q_k = \left\lceil \frac{t_k^*}{C} \right\rceil, \text{ where } t_k^* = \max_{i \in S_k} t_i \tag{8}$$

Equations (3) define all tasks in a stage. Equation (4) ensures that all the tasks are assigned. Equations (5) ensure that each task is assigned to only one stage. Inequalities (6) limit the total workload in a stage to the capacity of that stage. Conditions (7) enforce the precedence restrictions. Equations (8) enforce Buxey's practical consideration that, in a high tech assembly environment, parallel stations are only used when necessary to accommodate the largest task time [2].

3 HUP Algorithm

The HUP (Heuristic for U-shaped Parallel lines) algorithm belongs to the family of single pass, score, rank, and assign heuristics. It proceeds by repeatedly creating a new stage as required and filling the stage with the highest priority tasks until no tasks remain to be assigned. HUP uses a suite of scoring functions adapted from straight line balancing to calculate tasks priorities [5].

WE : work element time, the processing time of a task;

PW : positional weight, the sum of the processing times for a task and all the tasks that must follow (precede) it;

NF : number of followers, the number of tasks that follow (precede) a task;

NIF : number of immediate followers, the number of tasks that immediately follow (precede) a task.

Investigation determined that it was not possible to predict which the scoring function would provide the best solution for any particular network. Thus HUP's default mode is to apply the complete algorithm using each of scoring functions separately and choose the best of the resulting four solutions.

HUP organizes the information contained in the precedence network as a list of task objects with the following attributes:

t_k : processing time of task k;

$P1_k = \{i \mid (i,k) \in \{(i,j)\}\}$, set of immediate predecessors of task k;

$S1_k = \{j \mid (k,j) \in \{(i,j)\}\}$ set of immediate successors of task k;

$P2_k = P1_n \cup \bigcup_{p \prec n} P1_p$, set of all predecessors of task k;

$S2_k = S1_k \cup \bigcup_{s \succ n} S1_s$, set of all successors of task k.

We define the following variables:

q_n	the number of parallel stations in stage n,
$slack$	the slack time for the current stage,
T_n	the set of tasks assigned to stage n,
T_a	the set of task that have been assigned to any stage. Initialized to $\emptyset$.
A	the set of tasks available for assignment to the current stage. Initialized to $\emptyset$.

The following steps are repeated until all tasks are assigned.

Step 1: Create a new station n:

$$q_n = \emptyset,\ slack = C,\ (T_n) = \emptyset$$

Step 2: Look for tasks that have no unassigned predecessors or no unassigned successors by performing the following for each task $k \in T \cap T'_a$. Upon

completion go to Step 5

$$\text{if } P1_k \cap T'_a = \emptyset, \text{ perform step 3 and return}$$
$$\text{if } S1_k \cap T'_a = \emptyset, \text{ perform step 4 and return}$$

Step 3: Add the assigned task to A and calculate the ranking score for the predecessor free task k using the scoring function in effect.

$$A = A \cup \{k\}$$
$$\text{PW} : t_k + \sum_{i \in S2_k \cap T'_a} t_i \qquad \text{WE} : t_k$$
$$\text{NF} : \mid S2_k \cap T'_a \mid \qquad \text{NIF} : |S1_k \cap T'_a|$$

Step 4: Add the assigned task to A and calculate the ranking score for the successor free task k using the scoring function in effect.

$$A = A \cup \{k\}$$
$$\text{PW} : t_k + \sum_{i \in P2_k \cap T'_a} t_i \qquad \text{WE} : t_k$$
$$\text{NF} : \mid P2_k \cap T'_a \mid \qquad \text{NIF} : |P1_k \cap T'_a|$$

Step 5: Determine the task with the highest priority task which is feasible for the current stage. Priority is determined by the following boolean condition which defines task x less than task y:

$$x < y \equiv (rs_x < rs_y) \ \vee \ (rs_x = rs_y) \wedge (t_x < t_y)$$

Feasibility is determined by comparing the task time with the current slack time in the station. If the task time is not greater than the slack time then the task is added to the current stage. Special processing is done if the task time is greater than the cycle time. In such a case, the current stage is examined to determine if one or more parallel stations

should be added to the stage.

if $slack = 0$: go to step 10

loop :

if $A = \emptyset$: go to 10

Select the highest priority task i

if $t_i > C \ \wedge \ q = 1$: go to step 6

if $t_i < C$: go to step 7

Step 6: Determine if the current stage n (which consists of a single station) should be expanded to accommodate task i. The decision of whether to expand the stage incorporates Buxey's global constraint on minimizing duplicated equipment. Therefore, the limit to the number of parallel stations is $\lceil t_i/C \rceil$. This means that only one task i where $t_i > C$ may be contained in any stage. If the station cannot be expanded because of the time consumed by the tasks already assigned to the stage, the search for a feasible task continues.

$$q_{test} = \left\lceil \frac{t_i}{C} \right\rceil$$

$$t_{test} = slack + (q_{test} - 1)C$$

if $t_i \leq t_{test}$ then

$$q_n = q_{test}$$

$$slack = t_{test} - t_i$$

else go to 5

Step 7: Task i is assigned to the current stage and the set of tasks assigned to any station is updated:

$$T_n = T_n \cup \{i\},$$

$$T_a = T_a \cup \{i\}$$

$$slack = slack - t_i,$$

Step 8: The ranking scores for the tasks which remained in A are immediately adjusted for the effects of adding task i if the ranking criteria is PW, NF, or NIF. No adjustment is necessary for the WE ranking criteria. For all $k \in A$

$$\begin{aligned}
\text{PW:} &\quad \text{if } (i \in P2_k \ \vee \ i \in S2_k), \text{ then } rs_k = rs_k - t_i \\
\text{NF:} &\quad \text{if } (i \in P2_k \ \vee \ i \in S2_k), \text{ then } rs_k = rs_k - 1 \\
\text{NIF:} &\quad \text{if } (i \in P1_k \ \vee \ i \in S1_k), \text{ then } rs_k = rs_k - 1
\end{aligned}$$

Step 9: The predecessors and successors of task i just assigned are checked for availability.

$$\begin{aligned}
&\text{for all } k \in S1_i : \text{ if } P1_k \cap T'_a = \emptyset, \text{ perform step 3 and return} \\
&\text{for all } k \in P1_i : \text{ if } S1_k \cap T'_a = \emptyset, \text{ perform step 4 and return}
\end{aligned}$$

Go to step 5

Step 10: The current stage is completely loaded based upon the tasks that are currently available. Record the statistics of the current stage. If there are unassigned tasks, go to Step 1.

4 Computational Results

HUP was coded using C++ and run on a Pentium III 733Mhz computer. The benchmark data sets of Talbot, Hoffmann, and Scholl were used for evaluation. [6, 11, 12]. These data sets consist of 25 different networks. They have been used for testing and comparing solution procedures in almost all relevant studies since the early nineties.

There are no exact solvers for the U-shaped assembly line balancing problem where parallel stations are permitted. In order to obtain a basis for evaluation, the HUP algorithm was first compared with published solutions to non-parallel UALBP's based on the benchmark networks. The well known bin-packing lower bound for ALBP's, $LB = \lceil \sum t/C \rceil$ was also compared. There were 269 problems in this test set. The 269 problems took a total of 3 seconds or 0.011 seconds per problem to execute. The number of stations re-

sults are summarized in Table 1. In the table, deviation and relative deviation of solution s from x is defined as $s - x$ and $(s - x)/x$, respectively. HUP was then used to solve a set of problems each of which required at least one stage of two or more parallel stations. These problems were formed by varying the cycle time for each of the benchmark networks from the minimum task time to one less than the maximum task time of the respective network. Solutions were compared to the lower bound LB. There were a total of 14829 problems in this test set. The problems took 248 seconds or an average time of 0.0167 seconds per problem to run. The longest problem took 0.26 seconds. This corresponded to the somewhat unrealistic case where the largest network of 297 tasks was run with a very small cycle time resulting in nearly 14000 parallel stations. When the smallest cycle time was limited to 80% of the largest task time, the average time per problem was 0.010 seconds with the longest problem time of 0.07 seconds. The number of stations results for the parallel problems are summarized in Table 2.

Table 1: Summary for 269 non-parallel problems

	PW	NF	NIF	WE	Best of 4
Number equal to LB	42	44	45	46	59
Avg. Dev. from LB	2.02	1.98	1.90	1.88	1.80
Avg. Rel. Dev. from LB	0.089	0.087	0.087	0.087	0.076
Number equal to Optimal	108	111	111	117	133
Avg. Dev. from Optimal	0.75	0.71	0.64	0.62	0.53
Avg. Rel. Dev. from Optimal	0.044	0.043	0.042	0.042	0.032

Table 2: Summary for 14829 parallel problems

	PW	NF	NIF	WE	Best of 4
Number equal to LB	1494	1678	2378	2173	3038
Avg. Dev. from LB	7.10	6.93	6.76	6.67	6.36
Avg. Rel. Dev. from LB	0.069	0.065	0.062	0.062	0.055

5 Conclusions

It is clear that the HUP heuristic found the optimal solution to at least 20.4% of the 14829 parallel test problem cases, i.e., those cases for which the Hup

solution was equal to the lower bound LB. This correlates favorably with the non-parallel case where the heuristic had nearly 22% of its solutions equal to LB. For these non-parallel problems the optimal solution was found nearly 50% of the time. The average deviation from LB of the HUP solutions is larger for the parallel solutions because of the large values of stations resulting from small values of cycle time. The average relative deviations are an attempt to normalize the effect of large solution values. The average relative deviations from LB are actually much better than in the non-parallel case. If we assume this that the smaller average relative deviations of the parallel case implies that the relationship between LB's and optimal solutions for the parallel case is at least as good as the non-parallel case we could hypothesize that Hup finds the optimal solution on average $50^{+}\%$ of the time.

References

[1] I. Baybars. A survey of exact algorithms for the simple assembly line balancing problem. *Management Science*, 32(8):909–932, 1986.

[2] G. M. Buxey. Assembly line balancing with multiple stations. *Management Science*, 20:1010–1021, 1974.

[3] J. R. Freeman and J. V. Jucker. The line balancing problem. *Journal of Industrial Engineering*, 18:361–364, 1967.

[4] S. Ghosh and R. Gagnon. A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems. *International Journal of Production Research*, 27(4):637–670, 1989.

[5] S. Hackman, M. Magazine, and T. Wee. Fast, effective algorithms for simple assembly line balancing problems. *Operations Research*, 37(6):916–924, 1989.

[6] T. Hoffmann. Eureka: a hybrid system for assembly line balancing. *Management Science*, 38(1):39–47, 1992.

[7] R. M. Karp. Reducibility among combinatorial problems. *In Complexity of Computer Computations, (Proc. Sympos. IBM Thomas J. Watson*

Res. Center, Yorktown Heights, N.Y.).New York: Plenum,, pages 85–103, 1972.

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

[9] J. H. Salveson. The assembly line balancing problem. *Journal of Industrial Engineering*, 6(3):18–25, 1955.

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

[11] A. Scholl and R. Klein. Ulino: Optimally balancing u-shaped jit assembly lines. *International Journal of Production Research*, 37:721–736, 1999.

[12] J. Talbot, F. Patterson and W. Gehrlein. A comparative evaluation of heuristic line balancing techniques. *Management Science*, 32(4):430–454, 1986.