Balancing U-Shaped Assembly Lines with Parallel Stations

Louis J. Plebani
Lehigh University

Sihua Chen
Binney & Smith, Inc.

Report No. 04T-003

Balancing U-shaped Assembly Lines with Parallel Stations

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

The straight line assembly problem was introduced in 1955. Pressures of improved throughput and just-in-time manufacturing motivated development of models for straight lines with parallel stations in 1967 and U-shaped assembly lines in 1994. In this paper, we present a model for an assembly line balancing problem for U-shaped assembly lines where parallel stations are permitted and a best first branch and bound solver. The model incorporates the practical constraint that parallel stations are allowed only when necessary to accommodate a task with processing time greater than the cycle time. The solver includes a heuristic finishing procedure for cases were the solution time exceeds a maximum user specified timeout. We provide computational results on standard benchmark networks.

1. Introduction

The simple assembly line balancing problem type 1, consists of 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 the station cycle time, and the total number of stations in the line is minimized. The straight line assembly line balancing

problem was introduced by Salveson (1955) . In order to obtain increases in line efficiency, flexibility in assigning tasks into stations, and higher production rates, Freeman & Jucker (1967) suggested straight line balancing with parallel workstations. Buxey (1974) studied the practical aspects of parallel stations including costs of duplicated equipment and difficulties of layout and transportation. He concluded that parallel stations detract from the essential benefits of just-in-time flow line production and they should only be used to fit longer elements into demanded cycle time. Largely due to pressures of JIT, Miltenburg & Wijngaard (1994) introduced U-shaped line balancing and a heuristic solver. Sparling (1997) and Scholl & Klein (1999) developed branch and bound solvers for balancing single station U-shaped lines. This paper presents a best first branch and bound solver for a U-shaped assembly line balancing problem where parallel stations are permitted. In order to always provide a feasible solution, the solver provides the option of a heuristic finishing procedure if the solution time of the optimum solver exceeds a maximum user specified timeout.

2. Parallel station model

A U-shaped assembly line with parallel stations can be viewed as a U-shaped line of stages where any stage can be a single workstation or a group of two or more identical workstations operating in parallel. The effective cycle time for the workstation(s) in the k^{th} stage is $q_k C$, where q_k is the number of workstations in the stage and C is the system cycle time. The parallel station model can be obtained from Miltenburg & Wijngaard's (1994) model by introducing the variables q_k and changing the objective function to

$$\text{Minimize} \sum_k q_k, \tag{1}$$

changing the station capacity constraints to

$$\sum_{t_i \in S_k} t_i \leq q_k C, \tag{2}$$

and by constraining the number of stations in a stage to reflect Buxey's practical constraint that parallel stations should only be used to accommodate tasks whose task time is larger than the cycle time

$$q_k = \left\lceil \frac{t_k^*}{C} \right\rceil, \text{ where } t_k^* = \max_{i \in S_k} t_i. \tag{3}$$

Plebani & Chen (2003) created a heuristic solver called HUP for this problem.

3. Branch and bound solver

POBUL (Parallel Optimal Balancing for U-shaped Line) is a best first branch and bound solver having the general flow shown in figure 1. The major data structure is a priority queue of partial solutions. The partial solution represents the accumulation of tasks that have been assigned to some number of initial stages in the U-line. Each partial solution has an associated lower bound which is calculated considering the effects of the tasks already assigned. High priority in the queue is determined by smaller lower bound first, larger number of stations second, and smaller total slack third. Lower bound calculations use the well known bin packing lower bound $LB = \sum t_k / C$, which depends only on the cycle time C and the total task time $\sum t_k$. The initial upper bound is calculated using Plebani and Chen's HUP algorithm.

3.1 Branching algorithm

The branching algorithm operates on a partial solution which has been popped from the priority queue. The popped solution is discarded if its lower bound is greater than or equal

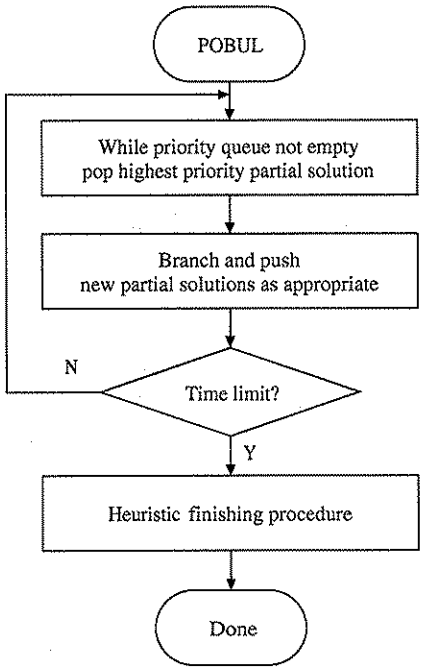

Figure 1: Basic POBUL Flow

to the current upper bound. The branching algorithm:

- determines all loads, i.e., combinations of the feasible tasks, which would fill a new stage,
- expands the new stage, if required, by adding parallel stations,
- adds all new partial solutions, which pass the fathoming checks, to the priority queue.

The procedure for determining loads is a U-shaped line adaptation of the backtracking procedure originally used by Hoffmann (1963) for his straight line heuristic algorithm. A vector of tasks av is used to store all tasks available for assignment to the present stage. A task is available if it has no unassigned predecessors and/or no unassigned successors. The vector av is sequentially searched for a task that can fit into the current stage, based upon slack time remaining. Special processing, as described below, is done if the task time is greater than the cycle time. When a task is found, it is assigned to the current load.

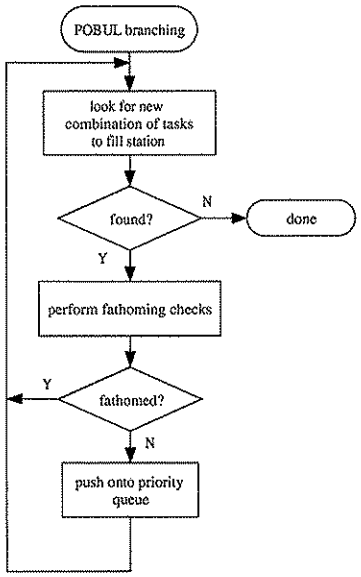

Figure 2: POBUL Branching

Its immediate predecessors and its immediate successors are checked for availability. If any have become available they are appended to av. When the end of av is reached, a load has been constructed which is then submitted to the fathoming checks. Additional loads are determined by backing up the search to the last task assigned. The task is removed from the current load and marked as being unavailable. The search continues from that point. All loads for the current stage have been determined when all tasks in av have been marked as unavailable. The maximum load rule discards any load that has labeled any task unavailable which could fit in the remaining slack for the load.

Special processing is done if a task time t is greater than the cycle time C. The task necessarily requires that its containing stage consist of parallel stations. The immediate decision is whether the current stage represented by the current load should be expanded to accommodate the task or should the task be considered infeasible for the current load and incorporated into later loads. The decision is made in the context of Buxey's (1974) practical constraint that the stage be expanded only if the expansion results in no more than $q = \lceil t/C \rceil$ stations in the stage. If the task can be incorporated, the number of parallel

stations in the stage is set equal to q, the slack time is adjusted accordingly, and the search to complete the load continues.

Each load that is incorporated into the current partial solution forms a new branch and bound node. Before a new node can be added to the branch and bound tree, by pushing it onto the priority queue, it must pass a series of fathoming checks.

The total slack rule compares the total slack of the partial solution to a test slack value computed from the current upper bound solution. The test slack value is the maximum slack that a solution could have and still result in one less station than the current upper bound. It is computed as

$$ts = (n_{UB} - 1) * C - \sum t_k \tag{4}$$

where n_{UB} is the number of stations in the current upper bound solution. If the slack of the current partial solution is greater than ts, the node is fathomed.

The equivalent solution rule avoids inclusion of equivalent partial solutions containing the same set of tasks but differing in the sequence of task assignment. The current partial solution is compared to the partial solutions already contained in the priority queue at the same depth in the branch and bound tree. If a match is found, the solution is fathomed.

As a requirement for entry on the priority queue, a lower bound for the node is computed by adding the bin packing lower bound calculated over the unassigned tasks to the stations contained in the partial solution. If this lower bound is greater than or equal to the current upper bound, the node is fathomed.

3.2 Heuristic Finishing Procedure

Because the assembly line balancing problem is NP-hard (Karp 1972), and to insure that the solver always returns a feasible solution, POBUL provides for a heuristic finishing procedure if the solution time exceeds a user specified timeout value. The heuristic procedure selects

nodes from the branch and bound tree and applies Plebani and Chen's HUP heuristic to the unassigned tasks in order to complete the solution. The best solution is reported.

4. Computational experiments

POBUL was coded using C++ and run on a 1 Ghz Pentium III PC. The benchmark precedence networks of Talbot & Gehrlein (1986), Hoffmann (1992), and Scholl & Klein (1999) were used for evaluation. These networks have been established as standard benchmarks in the literature and have been used for testing and comparing solution procedures in almost all relevant assembly line balancing studies since the early nineties. The Talbot data set is based on 12 precedence networks with 8-111 tasks. The Hoffman data set uses a subset of networks in the Talbot data set with 30-111 tasks. The Scholl data set is composed of precedence networks unique from the Talbot and Hoffman data sets with 25-297 tasks. Full descriptions of the precedence networks can be found in Scholl (1999)

To create the combined data set used for testing, the cycle time for each of the aforementioned precedence networks was varied from 20 percent of the maximum task time to one less than the maximum task time, thus ensuring at least one parallel station in each solution. All problem instances (with duplicates removed) form the combined data set with 12,161 instances. The timeout value was set to 10 minutes. If timeout occurs, the heuristic finishing procedure provided an upper bound UB on the optimal number of stations. Average results are summarized in table 1 which contains the following information:

# Instances:	number of problem instances for the dataset or precedence network;
# Optimum:	number of instances for which an optimal solution is found;
Av.rel.dev.:	average relative deviation for the data set (in parenthesis: average relative deviation for those instances which timed out);
Av.Cpu(s):	average computation time in seconds (includes time out of 600s).

The quality of the solutions provided by the heuristic finishing procedure was measured

by the relative deviation $(UB - LB)/LB$ of UB from the lower bound LB. Deviations are computed from LB because there are no known solutions to the problems, beyond the present work. Table 1 shows that POBUL found the optimal solution for 45 percent of all

Table 1: POBUL results for all problems

Data set	# Instances	# Optimum	Av.Rel.Dev.	Av.cpu(s)
Combined	12161	5453	0.0369(0.0535)	342.7
Hoffman	7648	2415	0.0410(0.0580)	428.6
Talbot	7815	2582	0.0407(0.0580)	419.5
Scholl	4346	2871	0.0300(0.0376)	204.7

problem instances. This ignores any cases where the heuristic finishing procedure produced an optimal solution not equal to LB. This percentage is a disproportionately influenced by some networks because the number of problems solved for each network was significantly different due to the manner in which the complete data set was constructed by varying the cycle time over the upper 80 percent range of the largest task time. Table 2 contains the test problem results for each of the networks. If each network is weighted equally, the percentage of optimal solutions for the combined set is nearly 70 percent.

The results show that POBUL parallels the performance of most reasonable algorithms for NP-hard problems in that a large portion of problems are solved within acceptable computational limits while others are somewhat computationally intractable in terms of available computer resources. Measures of network complexity, in addition to problem size, that have been proposed in the literature were examined as indicators of whether or not optimum solution of a particular problem would be successful. These included order strength (Mastor 1970), time interval $[t_{min}/c, t_{max}/c]$, and the time variability ratio $[t_{max}/t_{min}]$. These measures when combined with the number of tasks did not provide additional insight into problem complexity, as measured by percent of optimum solutions, than the number of

tasks alone. The major problem with these simple numerical measures is that each of them measures only a part of the problem characteristics which may influence problem difficulty. The interrelationships between the various elements of problem structure, in particular, the task time distributions assigned to resulting stages in the solution are not considered.

5. Summary and Conclusions

This paper introduced a new topic in assembly line balancing, optimal solution of single model U-shaped line balancing with parallel stations. A best first branch and bound optimal solver was presented together with computational results on a standard assembly line balancing test set. The solver was found the optimal solution approximately 70 percent of the sample networks. Traditional simple measures of problem complexity could not significant insight into computation difficulty. To estimate computational difficulty more complex measures which incorporate the structure of the precedence network and the corresponding time distributions of candidate tasks for each station are needed.

References

Buxey, G. M. (1974), 'Assembly line balancing with multiple stations', *Management Science* **20**, 1010–1021.

Freeman, J. R. & J. V. Jucker (1967), 'The line balancing problem', *Journal of Industrial Engineering* **18**, 361–364.

Hoffmann, T. (1963), 'Assembly line balancing with a precedence matrix', *Management Science* **9**(4), 551–562.

Hoffmann, T. (1992), 'Eureka: a hybrid system for assembly line balancing', *Management Science* **38**(1), 39–47.

Karp, R. M. (1972), 'Reducibility among combinatorial problems.', *In Complexity of Computer Computations, (Proc. Sympos. IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y.).New York: Plenum,* pp. 85–103.

Mastor, A.A. (1970), 'An experimental investigation and comparative evaluation of production line balancing techniques', *Management Science* **16**, 728–745.

Miltenburg, G. J. & J. Wijngaard (1994), 'The u-line balancing problem', *Management Science* **40**(10), 1378–1388.

Plebani, L. & S. Chen (2003), Hup: Heuristic for u-shaped parallel lines, Technical Report 03T-008, ISE Dept., Lehigh University, Bethlehem, PA.

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

Scholl, A. (1999), *Balancing and Sequencing of Assembly Lines*, 2nd edn, Springer Verlag.

Scholl, A. & R. Klein (1999), 'Ulino: Optimally balancing u-shaped jit assembly lines', *International Journal of Production Research* **37**, 721–736.

Sparling, D. (1997), Topics in U-line Balancing, PhD thesis, McMaster University, Canada.

Talbot, F. Patterson, J. & W. Gehrlein (1986), 'A comparative evaluation of heuristic line balancing techniques', *Management Science* **32**(4), 430–454.

Table 2: POBUL results by network

Network	#Tasks	#Instance	%Opt.	Av.rel.dev	Av.cpu(s)
Mertens	7	6	100.0	0.0820	< 0.1
Bowman	8	14	100.0	0.0746	< 0.1
Jaeschke	9	6	100.0	0.0750	< 0.1
Jackson	11	7	100.0	0.0243	< 0.1
Mansoor	11	37	100.0	0.0299	< 0.1
Mitchell	21	11	100.0	0.0301	< 0.1
Roszieg	25	12	100.0	0.0423	< 0.1
Heskiaof	28	86	100.0	0.0153	4.2
Buxey	29	21	100.0	0.0517	0.71
Sawyer	30	21	100.0	0.0476	3.57
Lutz1	32	1117	100.0	0.0552	< 0.1
Gunther	35	33	100.0	0.0273	0.55
Kilbridg	45	45	86.7	0.0054(0.0338)	86.0
Hahn	53	1411	100.0	0.0073	< 0.1
Warnecke	58	44	9.1	0.0655(0.0695)	562.2
Tonge	70	125	33.6	0.0286(0.0431)	404.2
Wee-Mag	75	23	0.0	0.2130(0.2130)	600.0
Arcus1	83	2929	25.4	0.0641(0.0820)	460.8
Lutz2	89	9	66.7	0.0246(0.0737)	252.7
Lutz3	89	61	70.5	0.0169(0.0368)	189.8
Mukherje	94	138	26.1	0.0229(0.0308)	451.4
Arcus2	111	4528	34.6	0.0267(0.0409)	413.8
Barthol2	148	67	0.0	0.0462(0.0462)	600.0
Barthold	148	303	60.4	0.0091(0.0231)	237.6
Scholl	297	1107	0.5	0.0342(0.0344)	598.9