Kimms, Alf

Working Paper

Balancing u-shaped assembly lines heuristically

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 524

Provided in Cooperation with:
Christian-Albrechts-University of Kiel, Institute of Business Administration

Manuskripte
aus den
Instituten für Betriebswirtschaftslehre
der Universität Kiel

No. 524

Balancing U–Shaped Assembly Lines Heuristically

A. Kimms

Manuskripte
aus den
Instituten für Betriebswirtschaftslehre
der Universität Kiel

No. 524

Balancing U–Shaped Assembly Lines Heuristically

A. Kimms

March 2000

Alf Kimms
Lehrstuhl für Produktion und Logistik, Institut für Betriebswirtschaftslehre, Christian–Albrechts–Universität zu Kiel, Olshausenstr. 40, 24118 Kiel, Germany
email: Kimms@bwl.uni–kiel.de
URL: http://www.wiso.uni–kiel.de/bwlinstitute/Prod
ftp://ftp.wiso.uni–kiel.de/pub/operations–research

Abstract

While balancing straight assembly lines is a well–studied problem, the situation where the assembly line has a U–shaped layout is a still emerging research field. The advantage of these lines is that workers may simply turn around to operate at two legs of the line. Hence, there is an additional degree of freedom for assigning operations to stations. This bears the hope that U–lines are more efficient than straight lines. Since work on this $\mathcal{NP}$–hard problem has mainly focused on exact algorithms, we study priority rule heuristics. We adapted 16 priority rules which have been applied to the straight line case before and examine their performance with a computational study.

Keywords: U–Lines, assembly line balancing, priority rule heuristics

1 Introduction

When operating an assembly (flow) line, one of the key performance measures is the efficiency of that line, and so, of course, we like to have lines with utmost efficiency. To define more precisely what efficiency means, we must take a closer look at the situation: We concentrate on paced lines on which a single (product) model is produced. Assembling such a model means to perform a certain set of operations. For technological reasons, these operations cannot be performed in arbitrary order, but have to respect acyclic precedence constraints, i.e., a partial order is given. While moving thru the line, a model subsequently passes several working areas, so–called stations, in each of which it remains a certain cycle time before it moves to the next station. Now, the balancing problem is to decide which operations are to be performed at what stations. It is not allowed to start an operation at one station and finish it at another station, i.e., each operation must be executed in full at one single station. However, it is allowed to assign more than one operation to a single station as long as the sum of operation times do not exceed the cycle time. Now, efficiency can more formally be defined as follows: Let V be the set of operations and t_j be the operation time of operation $j \in V$. Furthermore, let m be the number of stations and c be the cycle time. The efficiency E of a given assembly line is then defined as

$$E = \frac{\sum_{j \in V} t_j}{m \cdot c}.$$

Usually, either m or c are given a priori. As a consequence, maximizing E is equivalent to minimizing c, if m is given, or minimizing m, if c is given. If neither m nor c is given, one can compute an efficiency curve by fixing one of these variables and solving the remaining minimization problem. Doing this for several values of the variable that is fixed gives the desired curve. We focus here on the case that c is known in advance and consider this as being a realistic situation, because the cycle time c determines the production rate of the flow line which is tightly related to the demand that is to be met.

Traditionally, straight flow lines have been considered where an operation j can only be assigned to a station s if and only if all operations that must preceed j are assigned to stations not beyond s. A small example may illustrate this situation. Consider the data given in Figure 1 and assume a cycle time $c = 10$.

A feasible solution for this balancing problem is depicted in Figure 2. As we can see, five stations are required in this solution. Indeed, this is the minimum number of stations required.

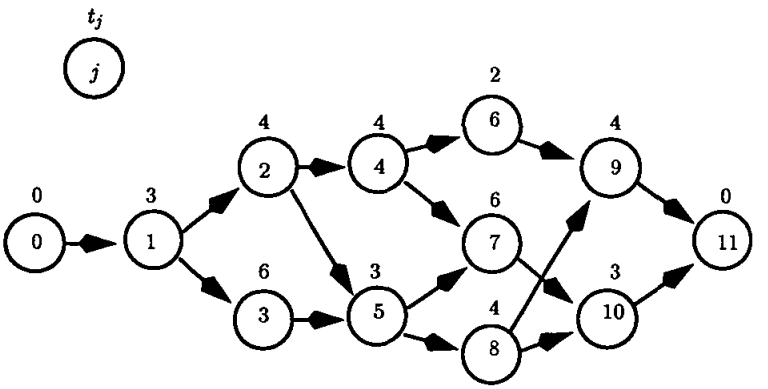

Figure 1: A Small Example

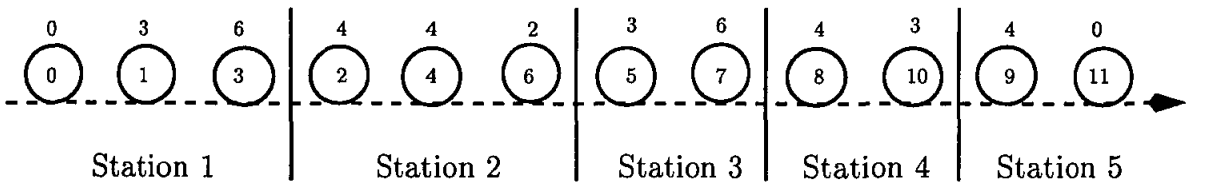

Figure 2: An Optimum Solution for the Straight Line

Instead of a straight layout of an assembly line a U–shaped layout has attracted attention in the recent past, because such formation may increase the efficiency of the line when compared to the straight line. The advantage of a U–line is that workers may now simply turn around to perform operations at different sections of the line. Hence, there is an additional degree of freedom for bundling operations to be performed at a common station. Figure 3 illustrates this situation. Apparently, only four stations are required now, and, therefore, this U–line is more efficient than the straight line.

There is a vast amount of literature on the assembly line balancing problem for straight lines. Some important references are the ones by Hoffmann [3], Johnson [4], Scholl and Klein [7], and Sprecher [8]. A much more comprehensive survey can be found in [6]. For U–shaped assembly lines, only few publications exist. A neat linear model formulation

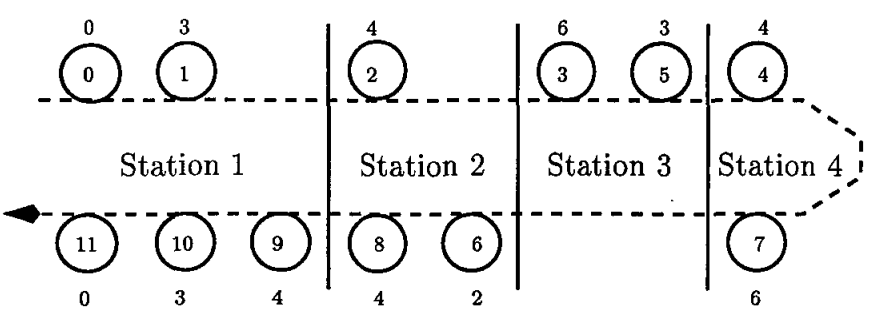

Figure 3: An Optimum Solution for the U–Line

with binary decision variables for the U–shaped assembly line balancing problem has been presented by Urban [10]. Methods for the U–shaped assembly line balancing problem are discussed by Miltenburg and Wijngaard [5] and Scholl [6].

In view of the literature, there seems to be a lack of contributions dealing with fast heuristics for the U–shaped assembly line problem. In what follows, we will therefore present several priority rule heuristics for the U–shaped assembly line balancing problem. Then, we discuss the results of a computational study.

2 Priority Rule Heuristics

For the straight assembly line balancing problem, a bunch of priority rule heuristics have been applied to find feasible solutions within short time. The performance of such heuristics has been evaluated and compared against each other in [6]. To sum up the results of this study, priority rule heuristics work very fast and, depending on the priority rules being employed, give solutions which deviate on average about 4% to 8% from the optimum result.

In this paper, we will adapt the heuristics used in [6] to attack the U–shaped assembly line balancing problem. Two basic construction principles can be discriminated for finding a feasible solution: station–oriented and operation–oriented construction schemes. Both construction schemes assign operations one by one to stations until a feasible solution is complete.

The station–oriented construction scheme works as follows: Starting with station number one as being the current station, we determine the set of all operations which can next be assigned to the current station. For being such a valid candidate, an operation must fulfill two criteria. First, its operation time plus the sum of operation times of all operations that have already been assigned to the current station must not exceed the cycle time. And second, all its predecessors or all its successors must have been already assigned to a station. If the set of operations fulfilling these two criteria contains more than a single element, a priority rule is used to make a choice. Afterwards, the set of valid candidates is updated to assign additional operations to the current station. Once the set of valid candidates is empty, the next (empty) station is considered and so on until all operations are assigned to a station.

The operation–oriented construction scheme works as follows: From the set of operations that have not been assigned to any station yet, we determine the set of operations which can next be assigned to a station. For being such a valid candidate, all its predecessors or all its successors must have already been assigned to a station. If this set contains more than a single operation, a priority rule helps to choose the next operation. Once an operation is selected, we assign it to the lowest numbered station such that two criteria are met. First, the sum of operation times of all operations assigned to a particular station must not exceed the cycle time. Second, an operation can only be assigned to a station s, if either all its predecessors have already been assigned to stations with no number greater than s, or all its successors have already been assigned to stations with no number greater than s. After an operation is assigned to a station, the set of valid candidates for the next assignment is updated and so on until a feasible solution is complete.

Throughout, we consider here single–pass heuristics only. That is, given a construction scheme and a priority rule, only a single feasible solution is constructed by making

deterministic choices. It is straightforward to extend these ideas to multi–pass heuristics where several runs are performed while making random choices.

In [6], 16 priority rules are tested. We have adapted all of them for our purposes:

Rule 1 (MinOpNo): A valid candidate j is selected by choosing the one with the lowest operation number

$$j.$$

This first rule is a very simple priority rule and we use it as a standalone rule, but also as a tie breaker rule for all other priority rules to identify a unique candidate.

For defining more sophisticated rules, we use V^- to denote the set of operations which have already been assigned to a station during the course of computation. Hence, V^- is updated after each selection of a candidate. Furthermore, for each operation j, F_j is the set of immediate successors, and P_j is the set of immediate predecessors. Likewise, F_j^* is the set of all successors, and P_j^* is the set of all predecessors. In the example above (see Figure 1), for instance, for $j = 5$ we have $F_5 = \{7, 8\}$, $F_5^* = \{7, 8, 9, 10, 11\}$, $P_5 = \{2, 3\}$, and $P_5^* = \{0, 1, 2, 3\}$. Note, if an operation j is a valid candidate then either $F_j^* \backslash V^-$ is empty or $P_j^* \backslash V^-$ is empty.

Rule 2 (MaxIFP): A valid candidate j is selected by choosing the one with the maximum number of unassigned immediate successors/predecessors, i.e. the one that maximizes

$$\max\{|F_j \backslash V^-|, |P_j \backslash V^-|\}.$$

Rule 3 (MaxFP): A valid candidate j is selected by choosing the one with the maximum number of unassigned successors/predecessors, i.e. the one that maximizes

$$\max\{|F_j^* \backslash V^-|, |P_j^* \backslash V^-|\}.$$

Priority rules which should somehow reflect information on the structure of the network of precedence constraints may utilize the number of arcs on paths having operation j as a source/sink. That is, they utilize

$$\vec{r}_j = |F_j \backslash V^-| + \sum_{h \in F_j^* \backslash V^-} |F_h \backslash V^-|$$

and

$$\overleftarrow{r}_j = |F_j \backslash V^-| + \sum_{h \in F_j^* \backslash V^-} |F_h \backslash V^-|.$$

Cumulated values have also been studied, i.e.

$$\vec{r}^*_j = \sum_{h \in F_j \backslash V^-} (\vec{r}^*_h + 1)$$

and

$$\overleftarrow{r}^*_j = \sum_{h \in P_j \backslash V^-} (\overleftarrow{r}^*_h + 1).$$

Rule 4 (MaxA): A valid candidate j is selected by choosing the one with the maximum number of arcs on paths to/from unassigned successors/predecessors, i.e. the one that maximizes

$$\max\{\vec{r}_j, \overleftarrow{r}_j\}.$$

Rule 5 (MaxCumA): A valid candidate j is selected by choosing the one with the maximum cumulated number of arcs on paths to/from unassigned successors/predecessors, i.e. the one that maximizes

$$\max\{\vec{r}^*_j, \overleftarrow{r}^*_j\}.$$

If priority rules should somehow reflect the workload that depends on operation j, one may use the operation time t_j as a measure. But, one may also use what are called the positional weights, i.e.

$$\vec{pw}_j = t_j + \sum_{h \in F_j^* \setminus V^-} t_h$$

and

$$\overleftarrow{pw}_j = t_j + \sum_{h \in P_j^* \setminus V^-} t_h.$$

Again, cumulated values have also been used, i.e.

$$\vec{pw}^*_j = t_j + \sum_{h \in F_j \setminus V^-} \vec{pw}^*_h$$

and

$$\overleftarrow{pw}^*_j = t_j + \sum_{h \in P_j \setminus V^-} \overleftarrow{pw}^*_h .$$

Rule 6 (MaxTime): A valid candidate j is selected by choosing the one with the maximum operation time

$$t_j.$$

Rule 7 (MaxPW): A valid candidate j is selected by choosing the one with the maximum positional weight, i.e. the one that maximizes

$$\max\{\vec{pw}_j, \overleftarrow{pw}_j\}.$$

Rule 8 (MaxCumPW): A valid candidate j is selected by choosing the one with the maximum cumulated positional weight, i.e. the one that maximizes

$$\max\{\vec{pw}^*_j, \overleftarrow{pw}^*_j\}.$$

From the positional weight, one can derive the earliest station away from the current station an operation can be assigned to. Formally, this is

$$E_j = \min\{\lceil \vec{pw}_j / c \rceil, \lceil \overleftarrow{pw}_j / c \rceil\}.$$

Since $|V|$ is an upper bound on the minimum number of stations required,

$$L_j = |V| - E_j$$

is the latest station an operation must be assigned to in order to improve that upper bound. Putting both together,

$$s_j = L_j - E_j + \epsilon$$

can be used as a measure for the slack (we add some small number ϵ, because s_j will later occur as a denominator and we want to avoid a division by zero).

Rule 9 (MinE): A valid candidate j is selected by choosing the one with the minimum offset, i.e. the one that minimizes

$$E_j.$$

Rule 10 (MinL): A valid candidate j is selected by choosing the one with the minimum latest station, i.e. the one that minimizes

$$L_j.$$

Rule 11 (MinSlack): A valid candidate j is selected by choosing the one with the minimum slack, i.e. the one that minimizes

$$s_j.$$

Combining the ideas of several of the above rules yields further priority rules. By combining rules 6 and 10 (or 11), we get two additional priority rules.

Rule 12 (MaxTimeL): A valid candidate j is selected by choosing the one with the maximum ratio

$$t_j/L_j.$$

Rule 13 (MaxTimeSlack): A valid candidate j is selected by choosing the one with the maximum ratio

$$t_j/s_j.$$

Combining rules 3 and 11 gives the following rule.

Rule 14 (MaxFPSlack): A valid candidate j is selected by choosing the one with the maximum ratio

$$\max\{|F_j^* \backslash V^-|/s_j, |P_j^* \backslash V^-|/s_j\}.$$

Combining rules 7 and (the reciprocal value of) 3 is also possible.

Rule 15 (MaxAvgPW): A valid candidate j is selected by choosing the one with the maximum ratio

$$\max\{\frac{\vec{pw}_j}{|F_j^*\backslash V^-|+1}, \frac{\overleftarrow{pw}_j}{|P_j^*\backslash V^-|+1}\}.$$

Finally, we combine rules 3 and 10.

Rule 16 (MinAvgL): A valid candidate j is selected by choosing the one with the minimum ratio

$$\max\{\frac{|V|-\lceil\vec{pw}_j/c\rceil}{|F_j^*\backslash V^-|+1}, \frac{|V|-\lceil\overleftarrow{pw}_j/c\rceil}{|P_j^*\backslash V^-|+1}\}.$$

3 Computational Study

We have implemented the priority rule heuristics in C and run the programs on a Pentium II computer with 330 MHz, 64 MB Ram and a Linux operating system. We have used the test sets that have previously been used by Scholl [6]. That is, we have used the 64 instances from the Talbot data set [9], the 50 instances from the Hoffmann data set[2, 3], the 168 instances from the Scholl data set [6], and a combined data set which includes all 269 instances (13 instances are in the Talbot data set as well as in the Hoffmann data set). Note that all of these instances have a unique dummy start and end operation, i.e. the first and the last operation has an operation time of zero. Because of this, we eliminate these two dummy operations in advance by assigning them to the first station at the very beginning. Otherwise, rules like MaxTime would produce the same result for U–lines as for straight lines which is not desired. Each of the 16 priority rules has been combined with the operation–oriented construction scheme (O) as well as with the station–oriented construction scheme (S). Note that for the straight assembly line balancing problem all rules are static (see [6]), i.e. the priority values need to be computed only once before the construction scheme takes over and updating is not necessary, but for U–lines these rules are dynamic rules (only MinOpNo and MaxTime remain static), because V^- changes during run–time which affects the priority values. The rules are implemented in a so–called immediate–update–first fashion (see, for instance, [1] or [6]), i.e. for each operation the priority value is updated immediately after each assignment of an operation to a station. As a measure of performance, we examined the (average and maximum) percentage deviation from the optimum objective function value (provided by Scholl [6]), the (average and maximum) absolute deviation from the optimum result, and the (average and maximum) run–time in CPU–seconds. Tables 1, 2, 3, and 4 provide these figures.

For most of the priority rules being employed, both construction schemes give similar results. By focusing on the combined data set (Table 4), we see that, if for a certain priority rule the operation–oriented construction scheme is superior, differences between both construction scheme are fairly small. On the other hand, if for a particular priority rule the station–oriented construction scheme is the better performer, then the differences are often significantly larger. This leads to the conclusion that the station–oriented heuristics should be preferred. In fact, Scholl [6] concluded that for straight assembly lines the station–oriented approaches are also "more promising" than the operation–oriented ones.

Not surprisingly, the run–time is small in (almost) all cases. Only the MaxCumA and the MaxCumPW priority rules resulted in excessive run–times longer than one hour when

	relative deviation				absolute deviation				run–time			
	AVG		MAX		AVG		MAX		AVG		MAX	
rule	O	S	O	S	O	S	O	S	O	S	O	S
MinOpNo	10.2	9.3	50.0	50.0	0.6	0.5	2.0	2.0	0.01	0.01	0.08	0.06
MaxIFP	8.1	7.9	50.0	50.0	0.5	0.5	2.0	2.0	0.02	0.01	0.12	0.09
MaxFP	6.2	7.2	50.0	50.0	0.4	0.4	2.0	2.0	0.02	0.01	0.12	0.07
MaxA	6.5	7.2	50.0	50.0	0.4	0.4	2.0	2.0	0.03	0.03	0.20	0.15
MaxCumA	6.9	7.1	50.0	50.0	0.4	0.4	2.0	2.0	0.05	0.04	0.34	0.29
MaxTime	6.2	5.0	33.3	33.3	0.4	0.3	2.0	1.0	0.01	0.01	0.09	0.08
MaxPW	5.8	6.2	50.0	50.0	0.3	0.4	1.0	1.0	0.02	0.01	0.11	0.07
MaxCumPW	5.1	6.2	50.0	50.0	0.3	0.4	1.0	1.0	0.04	0.04	0.30	0.26
MinE	9.8	9.8	50.0	50.0	0.6	0.6	3.0	3.0	0.02	0.01	0.11	0.07
MinL	9.7	9.4	50.0	50.0	0.5	0.5	2.0	2.0	0.02	0.01	0.12	0.08
MinSlack	9.7	9.4	50.0	50.0	0.5	0.5	2.0	2.0	0.02	0.01	0.12	0.08
MaxTimeL	6.2	5.0	33.3	33.3	0.4	0.3	2.0	1.0	0.02	0.01	0.12	0.08
MaxTimeSlack	6.2	5.0	33.3	33.3	0.4	0.3	2.0	1.0	0.02	0.01	0.12	0.09
MaxFPSlack	6.0	6.9	50.0	50.0	0.3	0.4	1.0	2.0	0.02	0.02	0.15	0.10
MaxAvgPW	4.9	4.5	33.3	33.3	0.3	0.3	1.0	1.0	0.02	0.01	0.11	0.07
MinAvgL	6.3	7.4	50.0	50.0	0.4	0.4	1.0	1.0	0.02	0.01	0.10	0.07

Table 1: Results for the Talbot Data Set

	relative deviation				absolute deviation				run–time			
	AVG		MAX		AVG		MAX		AVG		MAX	
rule	O	S	O	S	O	S	O	S	O	S	O	S
MinOpNo	9.9	7.8	20.0	20.0	1.5	1.0	4.0	2.0	0.03	0.02	0.09	0.06
MaxIFP	7.8	7.6	20.0	20.0	1.1	1.0	2.0	2.0	0.05	0.03	0.13	0.08
MaxFP	6.6	6.9	20.0	20.0	0.9	0.9	2.0	2.0	0.04	0.03	0.10	0.07
MaxA	6.8	6.9	20.0	20.0	0.9	0.9	2.0	2.0	0.09	0.06	0.21	0.14
MaxCumA	6.5	6.6	20.0	20.0	0.8	0.8	2.0	2.0	0.14	0.11	0.34	0.29
MaxTime	7.9	5.9	20.0	16.7	1.1	0.8	3.0	2.0	0.03	0.02	0.08	0.06
MaxPW	5.8	5.8	16.7	16.7	0.8	0.8	1.0	1.0	0.04	0.03	0.10	0.08
MaxCumPW	6.2	5.7	20.0	16.7	0.8	0.7	1.0	1.0	0.13	0.10	0.30	0.25
MinE	10.4	10.2	20.8	20.0	1.6	1.5	5.0	4.0	0.04	0.03	0.11	0.08
MinL	8.1	7.1	20.0	20.0	1.2	1.0	3.0	2.0	0.05	0.03	0.12	0.08
MinSlack	8.1	7.1	20.0	20.0	1.2	1.0	3.0	2.0	0.05	0.04	0.12	0.08
MaxTimeL	7.9	6.1	20.0	16.7	1.1	0.8	3.0	2.0	0.05	0.03	0.12	0.08
MaxTimeSlack	7.9	6.1	20.0	16.7	1.1	0.8	3.0	2.0	0.05	0.04	0.12	0.08
MaxFPSlack	6.3	6.7	20.0	20.0	0.8	0.8	2.0	2.0	0.06	0.04	0.15	0.09
MaxAvgPW	7.1	7.0	20.0	20.0	1.0	0.9	3.0	3.0	0.04	0.03	0.11	0.07
MinAvgL	6.8	6.7	20.0	20.0	0.9	0.8	2.0	2.0	0.04	0.03	0.11	0.07

Table 2: Results for the Hoffmann Data Set

rule	relative deviation AVG O	relative deviation AVG S	relative deviation MAX O	relative deviation MAX S	absolute deviation AVG O	absolute deviation AVG S	absolute deviation MAX O	absolute deviation MAX S	run–time AVG O	run–time AVG S	run–time MAX O	run–time MAX S
MinOpNo	7.9	7.4	33.3	33.3	1.8	1.7	9.0	9.0	0.29	0.22	1.67	1.21
MaxIFP	7.2	7.0	33.3	33.3	1.6	1.5	9.0	9.0	0.50	0.41	2.85	2.34
MaxFP	6.0	6.2	33.3	33.3	1.4	1.4	8.0	8.0	0.42	0.35	2.46	2.01
MaxA	6.0	6.2	33.3	33.3	1.3	1.4	8.0	8.0	1.37	1.11	8.11	7.20
MaxCumA*	6.6	6.8	33.3	33.3	1.4	1.4	6.0	6.0	0.40	0.34	1.16	1.01
MaxTime	6.6	4.5	33.3	33.3	1.5	0.9	6.0	4.0	0.30	0.22	1.74	1.22
MaxPW	5.2	5.4	33.3	33.3	1.1	1.1	7.0	7.0	0.42	0.35	2.42	2.00
MaxCumPW*	5.7	6.2	33.3	33.3	1.1	1.2	6.0	7.0	0.38	0.32	1.10	0.95
MinE	8.4	8.2	33.3	33.3	1.9	1.8	7.0	7.0	0.49	0.41	2.80	2.36
MinL	7.6	7.4	33.3	33.3	1.8	1.7	10.0	11.0	0.47	0.40	2.73	2.29
MinSlack	7.6	7.4	33.3	33.3	1.8	1.7	10.0	11.0	0.48	0.40	2.77	2.30
MaxTimeL	6.9	4.3	33.3	33.3	1.6	0.9	9.0	3.0	0.49	0.41	2.78	2.34
MaxTimeSlack	6.9	4.3	33.3	33.3	1.6	0.9	9.0	3.0	0.50	0.42	2.79	2.35
MaxFPSlack	6.3	6.5	33.3	33.3	1.4	1.5	9.0	9.0	0.52	0.42	2.84	2.34
MaxAvgPW	6.0	5.0	33.3	33.3	1.2	0.9	5.0	4.0	0.44	0.38	2.46	2.24
MinAvgL	5.9	6.2	33.3	33.3	1.3	1.4	8.0	8.0	0.43	0.36	2.45	2.03

*: These results are without the 26 instances with 297 operations from the Scholl data set, because the run–time per instance exceeds 1 hour when these rules are used.

Table 3: Results for the Scholl Data Set

rule	relative deviation AVG O	relative deviation AVG S	relative deviation MAX O	relative deviation MAX S	absolute deviation AVG O	absolute deviation AVG S	absolute deviation MAX O	absolute deviation MAX S	run–time AVG O	run–time AVG S	run–time MAX O	run–time MAX S
MinOpNo	8.7	7.8	50.0	50.0	1.5	1.3	9.0	9.0	0.19	0.14	1.68	1.28
MaxIFP	7.4	7.2	50.0	50.0	1.2	1.2	9.0	9.0	0.32	0.27	2.81	2.35
MaxFP	6.1	6.4	50.0	50.0	1.1	1.1	8.0	8.0	0.27	0.22	2.43	2.00
MaxA	6.1	6.4	50.0	50.0	1.0	1.1	8.0	8.0	0.90	0.70	8.44	7.11
MaxCumA*	6.5	6.7	50.0	50.0	1.0	1.1	6.0	6.0	0.27	0.23	1.14	1.01
MaxTime	6.5	4.7	33.3	33.3	1.2	0.7	6.0	4.0	0.20	0.14	1.74	1.20
MaxPW	5.3	5.5	50.0	50.0	0.9	0.9	7.0	7.0	0.28	0.23	2.45	2.02
MaxCumPW*	5.5	6.0	50.0	50.0	0.8	0.9	6.0	7.0	0.26	0.22	1.09	0.95
MinE	9.0	8.8	50.0	50.0	1.6	1.5	7.0	7.0	0.31	0.26	2.79	2.37
MinL	8.1	7.7	50.0	50.0	1.4	1.3	10.0	11.0	0.31	0.25	2.75	2.30
MinSlack	8.1	7.7	50.0	50.0	1.4	1.3	10.0	11.0	0.31	0.26	2.78	2.31
MaxTimeL	6.8	4.6	33.3	33.3	1.2	0.7	9.0	3.0	0.32	0.26	2.81	2.35
MaxTimeSlack	6.8	4.6	33.3	33.3	1.3	0.7	9.0	3.0	0.32	0.27	2.80	2.36
MaxFPSlack	6.1	6.5	50.0	50.0	1.1	1.1	9.0	9.0	0.34	0.27	2.81	2.33
MaxAvgPW	5.8	5.0	33.3	33.3	1.0	0.8	5.0	4.0	0.28	0.25	2.46	2.27
MinAvgL	6.1	6.5	50.0	50.0	1.0	1.1	8.0	8.0	0.28	0.23	2.44	2.02

*: These results are without the 26 instances with 297 operations from the Scholl data set, because the run–time per instance exceeds 1 hour when these rules are used.

Table 4: Results for the Combined Data Set

large instances with 297 operations are to be solved. Usually, the figures are fractions of a second.

On average, the feasible solutions require only less than two more stations than the optimum solution. The average percentage deviation from the optimum result is less than 10.4% in all cases. As one would expect, the simple rule MinOpNo gives worse results than most of the other rules. MinE, MinL, and MinSlack seem to be no good rules as well. MaxPW seems to be the top performer when an operation–oriented construction scheme is used showing an average deviation of 5.3% from the optimum result for the combined data set. For the station–oriented approach the rules MaxTimeL and MaxTimeSlack give best results with an average deviation of 4.6% only for the combined data set. Surprisingly, the simple rule MaxTime gives an almost equally good result (4.7% average deviation).

4 Conclusion

In this paper we have examined the performance of priority rule heuristics for the U–shaped assembly line balancing problem. 16 rules, which have previously been discussed in the context of straight assembly lines, have been adapted and applied. Each rule has been combined with an operation–oriented and a station–oriented construction scheme. As a test–bed we have taken instances from the literature. The main result is that station–oriented construction schemes outperform operation–oriented construction schemes. For the operation–oriented construction scheme, the best on average result has been achieved with a priority rule based on the so–called positional weight. This rule lead to an overall average deviation of 5.3% from the optimum result. for the station–oriented construction scheme, the best on average result has been achieved by using rules that combine the operation time with information on the latest station and the slack, respectively. These rules lead to an overall average deviation of 4.6% from the optimum result. This is a promising result given that we studied single–pass heuristics only. That is, this result could have been achieved by applying simple construction principles for finding a single feasible solution.

Future work should develop and test more sophisticated heuristics for balancing U–shaped assembly lines. Especially improvement heuristics can and should be considered.

Acknowledgement

Thanks to Michael Sülzer for implementing the heuristics in C. We are indebted to Armin Scholl for providing the optimum objective function values of the U–line instances.

References

[1] Hackman, S.T., Magazine, M.J., Wee, T.S., (1989), Fast, Effective Algorithms for Simple Assembly Line Balancing Problems, Operations Research, Vol. 37, pp. 916–924

[2] Hoffmann, T.R., (1990), Assembly Line Balancing: A Set of Challenging Problems, International Journal of Production Research, Vol. 28, pp. 1807–1815

[3] Hoffmann, T.R., (1992), Eureka: A Hybrid System for Assembly Line Balancing, Management Science, Vol. 38, pp. 39–47

[4] Johnson, R.V., (1988), Optimally Balancing Large Assembly Lines with FABLE, Management Science, Vol. 34, pp. 240–253

[5] Miltenburg, G.J., Wijngaard, J., (1994), The U–Line Balancing Problem, Management Science, Vol. 40, pp. 1378–1388

[6] Scholl, A., (1999), Balancing and Sequencing of Assembly Lines, Heidelberg, Physica, 2nd edition

[7] Scholl, A., Klein, R., (1999), Balancing Assembly Lines Effectively — A Computational Comparison, European Journal of Operational Research, Vol. 114, pp. 50–58

[8] Sprecher, A., (1999), A Competitive Branch–and–Bound Algorithm for the Simple Assembly Line Balancing Problem, International Journal of Production Research, Vol. 37, pp. 1787–1816

[9] Talbot, F.B., Patterson, J.H., Gehrlein, W.V., (1986), A Comparative Evaluation of Heuristic Line Balancing Techniques, Management Science, Vol. 32, pp. 430–454

[10] Urban, T.L., (1998), Optimal Balancing of U–Shaped Assembly Lines, Management Science, Vol. 44, pp. 738–741