An Exact Method for the Assembly Line Re-balancing Problem

Fatme Makssoud, Olga Battaïa, and Alexandre Dolgui

LIMOS, École des Mines de Saint-Étienne, France
{makssoud,battaia,dolgui}@emse.fr

Abstract. In this paper, we propose a mathematical optimisation model to solve the simple assembly line rebalancing problem. This problem arises when an existing assembly line has to be rebalanced in order to meet new production requirements. In this paper, a Mixed Integer Program is proposed for solving this problem with the objective to minimize the number of changes in the initial line. The computational experiments show the efficacy of the proposed method.

Keywords: Balancing and rebalancing of assembly lines, Binary integer programming.

1 Introduction and Related Literature

The assembly line consists of a number of consecutive workstations. Products are assembled by means of the successive execution of tasks in workstations as shown in Figure 1.

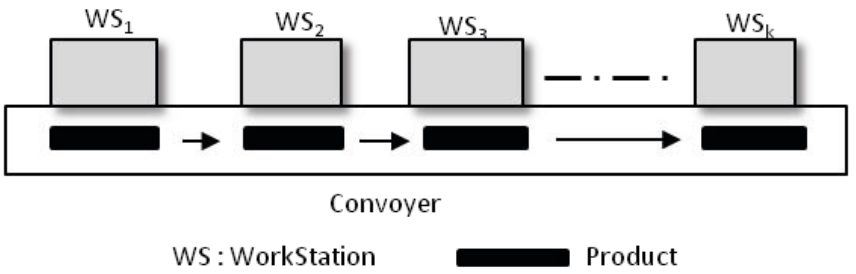

Fig. 1. Assembly line

The most known formulation of the Simple Assembly Line Balancing Problem (SALBP) aims to find a minimal number of workstations required for assigning a given set V of tasks taking into account precedence and cycle time constraints. The precedence constraints are given by a directed acyclic graph $G = (V, E)$ over

V. Prabhu, M. Taisch, and D. Kiritsis (Eds.): APMS 2013, Part I, IFIP AICT 414, pp. 159–166, 2013.

this set of tasks, where each edge $(i, j) \in E$ indicates that task i is an immediate predecessor of task j and therefore has to be assigned to a prior or the same workstation as task j.

Each task $j \in V$ is also characterized by its time, t_j. The sum of task times of the tasks assigned to the same workstation has to not exceed a given cycle time denoted by T_0. This problem is known to be NP-hard. Even if it was introduced in the literature almost 60 years ago [10], many recent studies still address it [2,3,7,9,11].

However, because of frequent changes in the product characteristics and demand, the problem that arises more frequently than initial line balancing problem is how to reassign the set of the tasks in order to meet new production requirements while minimize the number of modifications to be done in the initial line. We call this problem Simple Assembly Line Rebalancing Problem (SALReBP) and propose an exact method to solve it. Indeed, until now rebalancing problems for production lines have been principally addressed by means of approximate methods. Heuristics and genetic algorithms were used for solving stochastic assembly line rebalancing problem by Gamberini et al. [5,4]. For the case of a vehicle assembly line, three heuristic methods have been developed by Grangeon et al. [6]. A COMSOAL based heuristic for re-balancing of assembly lines that determines a fixed task sequence for a number of different cycle times was proposed by Agpak [1].

The remainder of this paper is organized as follows. A formal definition of the SALReBP is presented in the next section. The linearization of the model proposed is described in Section 3. An illustrative example is given in Section 4. A computational study is presented in Section 5 and concluding remarks are given in Section 6.

2 Formal Problem Definition and Mathematical Model

In this section, we present a formal definition of the SALReBP. As mentioned before, we denote by T_0 the cycle time and by m the number of workstations. Let V be the set of tasks to be allocated.

The following notations are used in our mathematical model:

- Indices:
  i, j for tasks,
  k for workstations.

- Parameters:
  V the new set of tasks, $j \in V$,
  t_j the processing time of task j, $j \in V$,
  $M = \{1, 2, ..., m\}$ is the set of workstations in the existing line,
  $L = \{1, 2, ..., l\}$ is the set of workstations in the new line, where l is an upper bound on the number of workstations for new line.
  $Q(j)$ is the interval of workstations in the upgraded line, where task $j \in V$

can be assigned. It is calculated using the precedence constraints.

– Decision variables:
$x^*_{jk} = 1$ if task $j \in V^* \subset V$ is assigned to workstation k in the initial configuration, 0 otherwise. Set V^* contains tasks j such that for $x^*_{jk} = 1, k \in Q(j)$; $x^*_{jk} = 0$ for all $k > m$, constraint 7 in model (1) - (7);

$y_{jk} = 1$ if task $j \in \cap V$ is assigned to workstation k in the new solution, 0 otherwise; $y_{jk} = 0$ for all $k \notin Q(j)$, constraint 6 in model (1) - (7).

The following model is used for the presented assembly line rebalancing problem. The objective function (1) consists in minimizing changes in the existing task assignment. Constraint (2) guarantees that every task j is assigned to one and only one workstation. Constraint (3) imposes the precedence constraints. Constraint (4) ensures that the total duration of the tasks assigned to workstation j does not exceed cycle time. Constraint (5) deals with the impossibility of executing certain tasks at the same workstation. Constraint (6) ensures that the variables outside intervals $Q(j)$ are set to 0.

$$Minimize \sum_{j \in V^*} \sum_{k \in L} \mid x^*_{jk} - y_{jk} \mid \tag{1}$$

$$\sum_{k \in Q(j)} y_{jk} = 1, \forall j \in V \tag{2}$$

$$\sum_{k \in L} k y_{ik} \leq \sum_{k \in L} k y_{jk}, \forall (i,j) \in A \tag{3}$$

$$\sum_{j \in V} t_j y_{jk} \leq T_0, \forall k \in L \tag{4}$$

$$y_{ik} + y_{jk} \leq 1, \forall \{i,j\} \in E, \forall k \in L \tag{5}$$

$$y_{jk} = 0, \forall j \in V, \forall k \notin Q(j) \tag{6}$$

$$x^*_{jk} = 0, \forall j \in V, \forall k > m \tag{7}$$

3 Linearization of the Model

In the following, we propose a method to linearize the proposed model.

Lemma 1. Let $x, y, z \in \{0, 1\}$. Then the following logical expression

$$\text{if } x = 1 \text{ and } y = 1, \text{ then } z = 1$$

can be modeled as follows:

$$x + y \leq z + 1.$$

Lemma 2. Let $x, y \in \{0, 1\}$. Then the following non-linear expression

$$z :=\mid x - y \mid$$

can be linearized using the following inequalities

$$x + y \leq (1 - z) + 1,$$
$$x + (1 - y) \leq z + 1,$$
$$(1 - x) + y \leq z + 1,$$
$$(1 - x) + (1 - y) \leq (1 - z) + 1,$$

It is evident to see that $z \in \{0, 1\}$. Moreover, only four following cases are possible:

$$\texttt{if } x = 1 \texttt{ and } y = 1; \texttt{ then } z = 0,$$
$$\texttt{if } x = 1 \texttt{ and } y = 0; \texttt{ then } z = 1,$$
$$\texttt{if } x = 0 \texttt{ and } y = 1; \texttt{ then } z = 1,$$
$$\texttt{if } x = 0 \texttt{ and } y = 0; \texttt{ then } z = 0.$$

Applying for these four cases Lemma 1, we obtain the necessary inequalities. To linearize the problem (1) - (5), we introduce a new variable $z_{jk} :=\mid x^*_{jk} - y_{jk} \mid$. Using Lemma 2, we obtain:

$$Minimize \sum_{j \in V^*} \sum_{k \in L} z_{jk} \tag{8}$$

$$\sum_{k \in Q(j)} y_{jk} = 1, \forall j \in V \tag{9}$$

$$\sum_{k \in L} k y_{ik} \leq \sum_{k \in L} k y_{jk}, \forall (i, j) \in A \tag{10}$$

$$\sum_{j \in V} t_j y_{jk} \leq T_0, \forall k \in L \tag{11}$$

$$y_{ik} + y_{jk} \leq 1, \forall \{i, j\} \in E, \forall k \in L \tag{12}$$

$$y_{jk} = 0, \forall j \in V, \forall k \notin Q(j) \tag{13}$$

$$x^*_{jk} = 0, \forall j \in V, \forall k > m \tag{14}$$

$$x^*_{jk} + y_{jk} \leq (1 - z_{jk}) + 1, \forall j \in V, \forall k \in L \tag{15}$$

$$x^*_{jk} + (1 - y_{jk}) \leq z_{jk} + 1, \forall j \in V, \forall k \in L \tag{16}$$

$$(1 - x^*_{jk}) + y_{jk} \leq z_{jk} + 1, \forall j \in V, \forall k \in L \tag{17}$$

$$(1 - x^*_{jk}) + (1 - y_{jk}) \leq (1 - z_{jk}) + 1, \forall j \in V, \forall k \in L \tag{18}$$

4 Illustrative Example

Let us consider the following case study. The initial assembly line is given in Figure 2. This line has to be rebalanced for a modified product where the following tasks {14, 19, 23, 28, 29} have been deleted and new tasks {31, 32, 33, 34, 35} have been introduced. The task times of all tasks are reported in Table 1. The precedence constraints to be respected are given in Figure 3. Exclusion constraints are:

{{1, 4}, {1, 17}, {1, 20}, {2, 11}, {3, 24}, {3, 7}, {4, 15}, {5, 22}, {6, 24}, {8, 21}, {9, 22}, {10, 15}, {11, 31}, {12, 13}, {12, 20}, {13, 28}, {15, 17}, {16, 17}, {22, 26}, {30, 33}, {31, 32}} and the cycle time $T_0 = 100$.

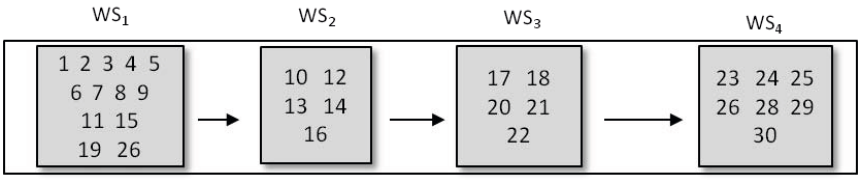

Fig. 2. Initial line

Table 1. Set of tasks V and their times

Task	Time (s)	Task	Time (s)	Task	Time (s)	Task	Time (s)	Task	Time (s)
1	0.93	7	0.68	13	0.64	21	0.78	30	0.91
2	1.06	8	0.16	15	0.09	22	0.64	31	0.72
3	0.68	9	0.68	16	0.17	24	0.09	32	0.15
4	0.16	10	0.16	17	0.09	25	0.17	33	0.19
5	0.68	11	1	18	0.12	26	0.09	34	0.33
6	0.16	12	0.78	20	1	27	0.12	35	0.97

The optimal solution was obtained in 0.36 second and consists to reassign the following tasks: {2, 4, 10, 12, 21, 31, 32, 33, 34, 35}, they are shown in bold in Figure 4 that illustrates the rebalanced line.

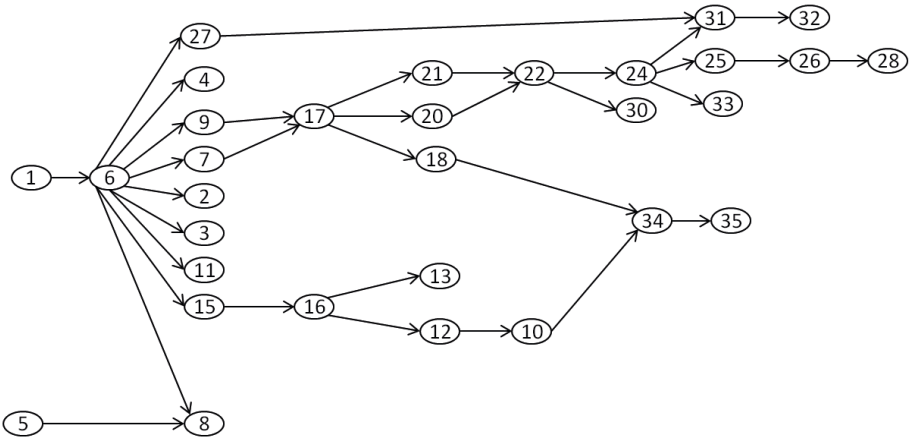

Fig. 3. The new precedence diagram

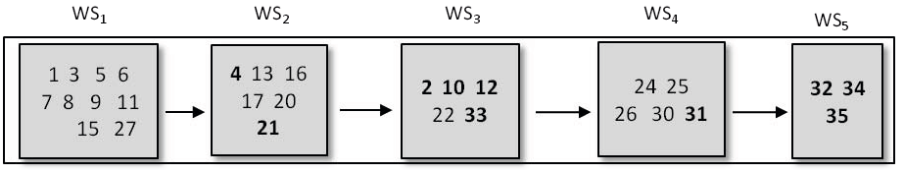

Fig. 4. Rebalanced line

Table 2. Results for rebalancing :25% Objective function

Instance number	tasks Changed	Time(s)	Instance number	tasks Changed	Time(s)	Instance number	tasks Changed	Time(s)
1	22	0.45	15	4	0.07	29	17	1.6
2	22	0.42	16	14	0.92	30	13	1.62
3	8	0.12	17	6	0.51	31	10	0.98
4	12	0.04	18	11	0.1	32	21	0.12
5	13	0.96	19	12	0.2	33	13	0.31
6	10	0.15	20	5	0.74	34	13	0.07
7	10	0.09	21	23	0.09	35	24	0.99
8	14	1.49	22	21	0.12	36	11	0.32
9	16	0.93	23	19	0.12	37	14	0.10
10	9	0.09	24	4	0.68	38	12	1.23
11	24	0.07	25	13	0.43	39	17	0.18
12	13	0.12	26	10	0.14	40	10	0.17
13	11	0.6	27	12	0.1	41	12	0.17
14	22	0.04	28	4	0.06	42	15	0.87

5 Computational Results

The proposed method was evaluated on a dataset that consists of 42 instance problems of 25 tasks each. For each initial problem, 3 versions of the modified product were generated: (1) by changing 25% of tasks (five tasks deleted and five added); (2) by changing 50% of tasks (five tasks deleted and eight added); (3) by changing 75% of tasks (five tasks deleted and fourteen added).

The computational results are respectively presented in Tables 2-4, where the values of the objective function indicating how many tasks were reassigned and

Table 3. Results for rebalancing :50% Objective function

Instance number	tasks Changed	Time(s)	Instance number	tasks Changed	Time(s)	Instance number	tasks Changed	Time(s)
1	23	0.2	15	6	0.45	29	21	0.51
2	27	0.54	16	14	0.18	30	13	0.18
3	14	1.04	17	5	0.29	31	10	1.07
4	13	0.4	18	15	0.2	32	20	0.24
5	21	0.29	19	12	0.07	33	15	0.14
6	12	0.51	20	5	0.18	34	16	0.12
7	11	0.12	21	23	1.31	35	24	0.29
8	14	0.37	22	20	0.14	36	15	0.14
9	18	0.67	23	20	0.23	37	15	0.18
10	12	0.92	24	4	0.21	38	15	0.31
11	24	0.1	25	12	0.1	39	18	0.2
12	14	0.2	26	11	0.09	40	10	0.17
13	14	0.59	27	13	0.17	41	13	0.6
14	22	0.18	28	10	0.98	42	17	0.53

Table 4. Results for rebalancing :75% Objective function

Instance number	tasks Changed	Time(s)	Instance number	tasks Changed	Time(s)	Instance number	tasks Changed	Time(s)
1	23	1.07	15	12	0.29	29	27	0.45
2	23	0.85	16	20	0.23	30	18	0.14
3	16	0.96	17	5	0.39	31	16	0.23
4	19	1.09	18	20	0.21	32	26	0.18
5	27	1.35	19	18	0.14	33	21	0.93
6	18	0.63	20	11	0.2	34	22	0.26
7	15	0.18	21	29	0.24	35	29	0.45
8	20	0.45	22	26	0.12	36	21	0.43
9	24	0.59	23	26	0.18	37	21	0.35
10	17	0.15	24	9	1.06	38	21	0.74
11	30	0.1	25	18	0.1	39	23	0.28
12	20	0.32	26	17	0.15	40	16	0.17
13	19	0.99	27	18	0.34	41	19	0.23
14	27	0.12	28	16	1.13	42	22	1.02

the solution time are given. Experiments were carried out on PC Intel(R), 2.20 GHz, with 8 Go RAM. The model was coded in C++ with ILOG CPLEX 12.4.

6 Conclusion

The problem of assembly line rebalancing was addressed. We have presented a mathematical optimization model that aims at minimizing the number of changes in the initial line. On the basis of an industrial case study, it was shown that the model proposed can be successfully applied in real world environment. The experimentation revealed that the model is capable of solving problems with up to 40 tasks in the precedence diagram.

The further research will undertake a more comprehensive computational experiment in order to evaluate the problem's size limit for which the exact method can be applied and to propose efficient heuristic or metaheuristic methods to obtain sub-optimal solutions for such problems.

References

1. Agpak, K.: An approach to find task sequence for re-balancing of assembly lines. Assembly Automation 30(4), 378–387 (2010)
2. Battaïa, O., Dolgui, A.: A taxonomy of line balancing problems and their solution approaches. International Journal of Production Economics 142, 259–277 (2013)
3. Bautista, J., Pereira, J.: A dynamic programming based heuristic for the assembly line balancing problem. European Journal of Operational Research 194(3), 787–794 (2009)
4. Gamberini, R., Gebennini, E., Grassi, A., Regattieri, A.: A multiple single-pass heuristic algorithm solving the stochastic assembly line rebalancing problem. International Journal of Production Research 47(8), 2141–2164 (2009)
5. Gamberini, R., Grassi, A., Rimini, B.: A new multi-objective heuristic algorithm for solving the stochastic assembly line re-balancing problem. International Journal of Production Economics 102(2), 226–243 (2006)
6. Grangeon, N., Leclaire, P., Norre, S.: Heuristics for the re-balancing of a vehicle assembly line. International Journal of Production Research 49(22), 6609–6628 (2011)
7. Kilincci, O.: Firing sequences backward algorithm for simple assembly line balancing problem of type 1. Computers &Industrial Engineering 60(4), 830–839 (2011)
8. Patterson, J.H., Albracht, J.J.: Technical NoteAssembly-Line Balancing: Zero-One Programming with Fibonacci Search. Operations Research 23(1), 166–172 (1975)
9. Pastor, R., Ferrer, L.: An improved mathematical program to solve the simple assembly line balancing problem. International Journal of Production Research 47(11), 2943–2959 (2009)
10. Salveson, M.E.: The assembly line balancing problem. Journal of Industrial Engineering 6(3), 18–25 (1955)
11. Sewell, E.C., Jacobson, S.H.: A branch, bound, and remember algorithm for the simple assembly line balancing problem. INFORMS Journal on Computing (2011), doi: 10.1287/ijoc.1110.0462