Notes on Complexity of the Simple Assembly Line Balancing Problem

Lazarev A.A., Gafarov E.R.
Institute of Control Sciences of the Russian Academy of Sciences, Profsoyuznaya st. 65, 117997 Moscow, Russia

Dolgui A.
Ecole Nationale Supérieure des Mines, FAYOL-EMSE, CNRS:UMR6158, LIMOS, F-42023 Saint-Etienne, France

Abstract

In this paper, we consider the assembly line balancing problem, for which it is necessary to minimize the number of used machine for a given cycle time. We propose a special case of the problem for which any Branch and Bound algorithm with any polynomial time computed Lower Bound can't solve some instances even for n=60 operations in appropriate time. Additionally, we analyze the worst maximal--station-load line balance and present a technique to reduce the graph of precedence relations that provides some advantages.

Introduction

We consider a single-model paced assembly line (see [Scholl] for definitions and terminology) which continuously manufactures a homogeneous product in large quantities (a mass production).
The simple assembly line balancing problem (SALBP-1) is to find an optimal line balance for a given cycle time c, i.e., to find a feasible assignment of the given operations to the stations in such a way that the number of used stations m reaches its minimal value. The SALBP-1 is defined as follows.

Given a set N={1, 2, …,n} of operations and K stations 1,2,…,K. Operations $j \in N$ has to be processed for $t_j \geq 0$ time units. The cycle time $c \geq \max\{t_i,\ j \in N\}$is given.
Furthermore, finish-start precedence relations $i \to j$ are defined between the operations according to an acyclic directed graph $G.$ The objective is to assign each operation j, j=1, 2, …,n, on a stations in such a way that:

- the number m <= M of used stations is minimized;
- for each station k = 1,2,…,m a total load time $\sum_{j \in N_k} t_j$doesn't exceed c, where N_k-- a set of operations assigned on the station k;
- the given precedence relations are fulfilled, i.e. if $i \to j,\ i \in N_{k_1}$and $j \in N_{k_2}$then $k_1 \leq k_2$.

The problem is NP-hard in the strong sense. Surveys about results for SALBP-1 are published periodically (e.g. [Scholl]). There exists a special electronic library http://www.assembly -line-balancing.de of experimental data to test solution algorithms for this problem. The best known Branch and Bound algorithm is presented in [Sewell].

The rest of the paper is organized as follows. In the first section we present a special case for which any Branch and Bound (B&B) algorithm with any polynomial time computed Lower Bound has exponential run time and the best known algorithm [Sewell] is not able to solve some instances of this special case even for n>=60 in an appropriate time. A simple assembly line balancing problem, where a number of used machines is maximized on solutions with maximal

station loads, is considered in Sections 2 and 3. A modification of a graph of precedence relations to a planar one and the benefits of such the modification are presented in Section 4.

1. Run Time of a Branch and Bound Algorithm for a Special Case

Partition problem. Given is a set $N = \{b_1, b_2, \ldots, b_n\}$ of numbers $b_1 \geq b_2 \geq \cdots \geq b_n > 0$ with $b_i \in Z_+,\ i = 1, 2, \ldots, n$, and a number $A \in Z_+$ with $A < \sum_{j \in N} b_j$. Is there a subset $N' \subset N$ such that $\sum_{j \in N'} b_j = A$?

The worst case analysis of a run time of $B\&B$ algorithms are presented, e.g., in [Finkelstein,Posypkin] for the well-known Knapsack Problems. In these papers authors use special cases, for which it's can be easy to find an optimal solution with a B&B algorithm, but to prove its optimality we should consider almost all feasible solutions. The number of optimal solutions for instances of the special cases

In [Posypkin], the following special case of the Knapsack Problem has been presented. For this type of instances, we have a set of numbers $B = \{b_1, b_2, \ldots, b_n\}$ which are used in the following parameterized optimization instance:

$$\begin{cases} f(x) = \sum\limits_{j=1}^{m+n} c_j x_j \to \max \\ \sum\limits_{j=1}^{m+n} c_j x_j \leq ka + 1, \\ a \geq 2, \\ c_j = a,\ j = 1, 2, \ldots, m, \\ c_j = b_j \cdot a,\ j = m+1, 2, \ldots, m+n, \\ x_j \in \{0, 1\},\ j = 1, 2, \ldots, m+n. \end{cases} \qquad (1)$$

For this special case any B&B algorithm with any Upper Bound computed in a polynomial time, has an exponential number of nodes in a search tree, if $P \neq NP$ [Posypkin], i.e. the complexity time of the B&B algorithm is exponential and equals to $\sim \frac{3}{2} \cdot \frac{2^{n+3/2}}{\sqrt{\pi(n+1)}}$ [Posypkin]. In fact, in an instance of this type, there is a sub-instance which is an instance of the NP-hard Partition Problem with numbers $B = \{b_1, b_2, \ldots, b_n\}$, i.e. to compute a "good" Upper bound we should solve an instance of the Partition Problem, that impossible to do in a polynomial time, if $P \neq NP$. As a consequence, the reduction of nodes by rule "a local upper bound <= current record" will be ineffective.

We use a similar idea to construct a special case of SALBP-1 for which any $B\&B$ algorithm with any polynomial time computed Lower Bound has an exponential time complexity which makes the algorithm ineffective for instances with n>=60 operations.

We use the following reduction from the Partition Problem to the special case of SALBP-1. Let's modified an instance of the Partition problem as follows:

Modified instance of the Partition problem. Given is a set $\overline{N} = \{\overline{b}_1, \overline{b}_2, \ldots, \overline{b}_{2n}\}$ of numbers $\overline{b}_1 \geq \overline{b}_2 \geq \cdots \geq \overline{b}_{2n} > 0$ with $\overline{b}_i \in Z_+,\ i = 1, 2, \ldots, 2n$, and a number $\overline{A} \in Z_+$ with $\overline{A} < \sum_{j \in \overline{N}} \overline{b}_j$. The numbers $\overline{b}_i,\ i = 1, 2, \ldots, 2n$ are denoted as follows:

$$\begin{cases} \overline{b}_{2n} = 1, \\ \overline{b}_{2i} = 2 \cdot \sum_{j=i+1}^{n} \overline{b}_{2j-1}, & i = n-1 \ldots, 1, \\ \overline{b}_{2i-1} = \overline{b}_{2i} + b_i, & i = n \ldots, 1, \end{cases} \qquad (2)$$

where $b_1, b_2, \ldots, b_n$ – numbers from the initial instance. Let $\overline{A} = \sum_{i=1}^{n} \overline{b}_{2i} + A$. The question is: "Is there a subset $\overline{N}' \subset \overline{N}$ such that $\sum_{j \in \overline{N}'} \overline{b}_j = \overline{A}$"?

So, any instance of the Partition problems can be in polynomial time transformed to an equivalent instance of the modified problem. If the initial instance has the answer "YES" (and the same answer has the modified instance) then $\overline{N'}$ contains one and only one number $\overline{b}^i$ from each pair $\{\overline{b}_{2i-1}, \overline{b}_{2i}\},\ i = 1, 2, \ldots, n$. If the number b_i is included in the set N' then $\overline{b}_{2i-1}$ is included in $\overline{N'}$, otherwise the number $\overline{b}_{2i} \in \overline{N'}$. Without lost of generality let us assume $\overline{A} = \frac{1}{2} \sum_{i=1}^{2n} \overline{b}_i$.

In the special case of SALBP-1 we have 2n operations and $c = \overline{A}$. Let $k = \min\{w | 10^w \geq 2A\}$. Processing times of operations are denoted as follows:

$$t_i = 10^k + \overline{b}_i,\ i = 1, 2, \ldots, 2n.$$

There are no precedence relations between operations.

It's obvious that if and only if the modified instance of the partition problem has the answer "YES" then the minimal number of machines m*=2, otherwise m*=3. As a consequence, if $P \neq NP$, there is no polynomial time computed Lower Bound with a relative error equal or less than 3/2. That means, for any set of polynomial time computed Lower Bounds $\{LB_1, LB_2, \ldots, LB_X\}$, there is an modified instance of the Partition problem with an answer "NO", for which $LB_x = 2,\ i = 1, 2, \ldots, X$, although m* = 3. For such an Partition instance any feasible solution of the corresponding SALBP-1 instance is optimal, but to prove its optimality we should consider almost all the feasible solutions.

Let's estimate the possible number of feasible solution. On the first machine there could be processed at least n-1 operations then there is at least $\binom{2n}{n-1}$ possible loads of the first machine, i.e. the number of feasible solutions is greater than

$$\binom{2n}{n-1} > \binom{2n}{n} \approx \frac{2^{2n}}{\sqrt{n\pi}}$$

To solve such the instance of SALBP-1 with 2n = 60 a computer must perform more than $\frac{2^{60}}{10}$ operations. Let's assume that the fastest known computer performs 2^{30} operations per second, or less than 2^{47} operations per day. Then a run time of an algorithm will be more than $\frac{2^{13}}{10} > 800$ days days! That means there are instances of SALBP-1 for which a B&B algorithms with polynomial time computed lower bounds have a too big run time.

Let's us consider the best known B&B algorithm from the paper [Sewell]. This algorithms solves all benchmark instances published online on http://www.assembly
-line-balancing.de in less than one second per instance. We analyze some of methods which are used in the algorithm to reduce the search tree.

1. Lower bounds. $LB_1 = \lceil \frac{\sum t_i}{c} \rceil$, $LB_2 = |\{j \in N | t_j > c/2\}| + \lceil \frac{|\{j \in N | t_j = c/2\}|}{2} \rceil$. LB_3 is computed by assigning a weight w_j to each operation j:

$$w_j = \begin{cases} 1 & if t_j > 2c/3, \\ \frac{2}{3} & if t_j = 2c/3, \\ \frac{1}{2} & if c/3 < t_j < 2c/3, \\ \frac{1}{3} & if t_j = c/3. \end{cases} \quad (3)$$

then $LB_3 = \lceil \sum w_j \rceil$. For the considered instance of SALBP-1 LB1=2 and $LB_2 = LB_3 = 0$. It is obvious that all of these Lower Bounds are useless for the proposed special case.

2. In the algorithm, to solve subproblems a B&B algorithms for the Bin-packing problem is used. For the considered instance such the algorithm for the Bin-packing problem has an exponential run time as well, i.e. with such technique the search tree will not be reduced substantially.

3. Authors use a modification of breadth-first strategy of branching, which could be unappropriated for such kind of instances in cause of big demand to memory.

We can conclude the following. Despite the algorithm [Sewell] solves all benchmark instances in less than 1 second per instance, known B&B algorithms for SALBP-1 remains exponential and can't solve some instances with the size n>60 in appropriate time. That's why we consider a work in the field of exact algorithms for the general case of the problem unpromising. Researcher can concentrate on special cases or on essentially new solution schemes.

2. Maximization of Number of Stations

To propose an essentially new solution scheme for SALBP-1, it is necessary to investigate properties of optimal solutions. We can investigate not only properties of good solutions but properties of worst solutions as well, to avoid solutions with such the properties. In this Section in contrast to classical SALBP-1, where the number of used stations should be minimized we consider an optimization problem with opposite objective criteria, namely we consider the maximization of the number of stations.

To make the maximization problem not trivial we assume that all stations (instead the last one) should be maximal loaded, i.e. for two stations $m_1, m_2,\ m_1 < m_2$ there is no operation j assigned on the station m_2 which can be assigned on station m_1 without violation of precedence constraints or the feasibility's condition "total load time of the station doesn't exceed the cycle time". Such solutions with maximal station loads we call {\em active} solutions.

Usually in solution algorithms or algorithms of computing of Upper bounds, such active solutions are considered. Let m -- a number of stations for a feasible solution with maximal station loads, m^{min} -- a minimal number of stations and m^{max}-- a maximal number. It is known [Hackman] that $\frac{m}{m^{min}} < 2$ and as a consequence $\frac{m^{max}}{m^{min}} < 2$.

Denote the maximization problem by max-SALBP-1.

On the one hand, the investigation of a particular problem with the {\it maximum} criterion is an important theoretical task. Algorithms for such a problem with the maximum criterion can be used

to cut bad sub-problems in the branching tree of branch-and-bound algorithms, to compute Upper and Lower Bounds for a bi-criterion problems [Aloulou2010]. On the other hand, such problems have also practical interpretations and applications [Aloulou2004,Arkin,Gafarov]. Moreover a situation arises when the company is considered as a customer, and one wants to know the worst variant of a line balance, which is computed in a `black box' (e.g. in a plant).

Lemma 1. max-SALBP-1 is NP-hard in the strong sense
Proof. By reduction from 3-Parition Problem

Lemma 2. max-SALBP-1 is not approximated with an approximation error 3/2, unlike P = NP.
Proof. By reduction from Parition Problem to the special case where m* = 2.

In [Queyranne], it was proved that for SALBP-1 any polynomial time heuristic has worst-case ratio at least 3/2, i.e. there for a heuristic algorithm there is an instance for which $\frac{m}{m^{min}} \geq \frac{3}{2}$. If we consider SALBP-1 as a generalization of the Bin-Packing Problem (BP), then this approximation result can be compared with known results for the BP (e.g., see [Queyranne]), where are a heuristic, for which $m \geq 11/9m^{min}+1$. For BP it's known that any polynomial time heuristic has worst-case ratio at least 3/2, as well. But it's hold only for $m^{min}=2, m=3$, i.e. the absolute error equals 1. For $m^{min}>2$we have a worst case $m > 11/9m^{min}+1$. We can conjecture, that the same relation holds for SALBP-1, too, but in [Queyranne], authors show that worst-case ratio 3/2 holds for any absolute error, i.e., for any polynomial time heuristic, for any $m^{min} \geq 2$, there is an instance for which $\frac{m}{m^{min}} = \frac{3}{2}$.

Next we prove a similar result for max-SALBP-1.
Lemma 3. For any polynomial time heuristic for max-SALBP-1 , for any given absolute error $q \geq 3$, there is an instance for which $\frac{m^{max}}{m} = \frac{3}{2}$ and $m^{max} - m = q$.

3. Maximal Number of Stations for Benchmark Instances

In this section we propose some experimental results on benchmark instances. Although, it's obvious that there are instances for which $\frac{m^{max}}{m^{min}} = \frac{3}{2}$(e.g., see Lemma 2), and there are instances for which $\frac{m^{max}}{m^{min}} \approx 2$ [Scholl], it's interesting to compare the numbers on benchmark instances. The goal of the experiments is to estimate the maximal number of station for some of benchmark instances presented on http://www.assembly-line-balancing.de. To maximize the number of machines we constructed a simple B&B, with deep-first branching strategy and a simple Upper Bound which are based on the following observation:

We compute $Pred_j$-- a set of all predecessors (not immediate, too) for an operation j=1,2,…,n. Then the minimal number of the machine S_j, to which the operation j=1,2,…,n, can be assigned, is computed as $S_j = \lceil \frac{\sum_{in \in Pred_j} t_i + t_j}{c} \rceil$. Denote $ML_k,\ k = 1, 2, \ldots, n$ is a minimal load of the station k. Let $t_k^{min} = \min_j\{t_j | S_j \leq k\}$ and $t_k^{max} = \max_j\{t_j | S_j \leq k\}$. Then $ML_k = \max t_k^{min}, c - t_k^{max} + 1$.

Some other trivial Upper Bounds are used, e.g., $UB = \lceil \frac{2 \cdot \sum t_j}{c} \rceil - 1$. Upper bounds are recomputed for each subproblem.
Unfortunately, the proposed B&B can't solve the majority of benchmark instances in 10 minutes on CPU INTEL CORE 2 DUO 2,4 Hz (only one processer is used), but the received results allow us estimate the difference between the minimal number of machine and possible maximal number.

In the following table the experimental results are presented:

Instance	n	c	m^{min}	m^{max}	run time (sec)
Arcus1	83	3786	21	24	600,00
Arcus2	111	5755	27	30	600,00
Barthol2	148	84	51	57	600,00
Barthold	148	403	14	16	600,00
Bowman	8	20	5	5	0,00
Buxey	29	27	13	16	600,00
Gunther	35	41	14	16	600,00
Hahn	53	2004	8	9	600,00
Heskiaoff	28	138	8	10	600,00
Jackson	11	7	8	9	0,00
Jaeschke	9	6	8	8	0,00
Kilbridge	45	56	10	12	600,00
Lutz1	32	1414	11	13	600,00
Lutz2	89	11	49	52	600,00
Lutz3	89	75	23	25	600,00
Mansoor	11	48	4	5	0,00
Mertens	7	6	6	6	0,00
Mitchell	21	14	8	10	0,36
Mukherje	94	176	25	27	20,90
Roszieg	25	14	10	11	247,32
Sawyer	30	25	14	17	600,00
Scholl	297	1394	50	54	600,00
Tonge	70	160	23	26	600,00
Warnecke	58	54	31	36	600,00
Wee-mag	75	28	63	63	14,66

If the run time less than 600 then the presented m^{max} is optimal. After 60 seconds or running of the algorithm, we had the same values m^{max} for all the instances, except Arcus2 (after 60 sec. m^{max}= 29) and Lutz1 (after 60 sec. m^{max}= 12). The maximal founded deviation $m^{max} - m^{min}$ doesn't exceed 20%.

4. Flat Graph of Precedence Relations

In [Lazarev] authors propose a transformation of graph with precedence constraints to a planar graph for the well-known Resource-constrained Project Scheduling Problem. The same idea can be used for SALBP-1.

Theorem 1. For any instance of SALBP-1 with n operations and v precedence relations, there exists an analogous instance with a flat graph G' with n' operations and v' relations, where n + v > n'+v'.

We obtain an analogous instance from the original one by adding "dummy" operations (with t_j = 0) and deleting all the unnecessary relations.
The proof of the theorem follows from Lemmas 4 and 5.

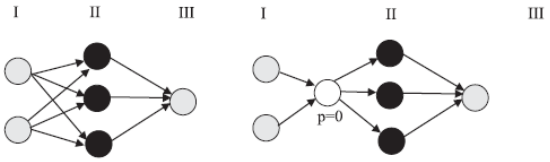

Figure 1: Transformation of $K_{3,3}$

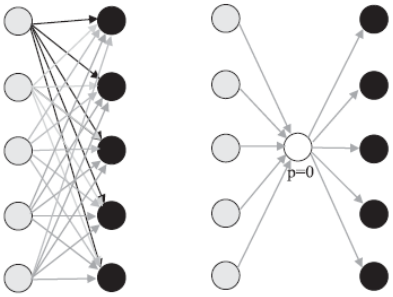

Figure 2: Transformation of $K_{k,k}$

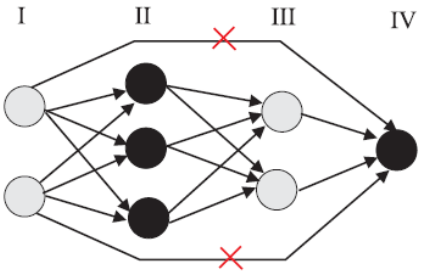

Figure 3: Transformation of $K_{4,4}$

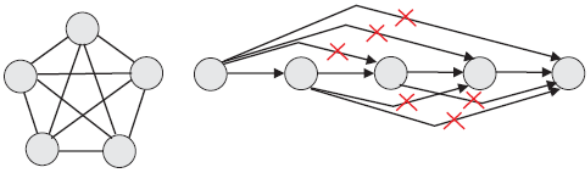

Figure 4: Transformation of $K_{5,2}$

Lemma4. If there is a subgraph $G' \subset G$ that is isomorphic to the special graph $K_{3,3}$, then we can transform it into a flat subgraph by adding "dummy" jobs (with $t_j = p = 0$) and deleting all the unnecessary relations.

See the transformation rules shown in Figures 1,2.3.

Lemma5. If there is a subgraph $G' \subset G$ that is isomorphic to the special graph $K_{5,2}$, then we can transform it into a flat subgraph by deleting all the unnecessary relations.

See the transformation rule shown in Figure 4.

The number of precedence relations influences on the run times of solution algorithms. Complexity time of algorithms (including algorithms of calculating of lower and upper bounds) the number of precedence relations is estimated by different authors as $O(n^2)$(i.e, the "Order strength" on http://www.assembly-line-balancing.de is estimated according to the number n(n-1) of precedence relations). If we consider only instances with a planar graph then the number of relations is <=3n-6, i.e. $O(n)$.

So, the fact mentioned in Theorem 1 allows us reduce the run time of algorithms (by reduction of unnecessary relations) and estimate the complexity exacter.

Conclusion

In this paper, we propose some complexity results for SALBP-1. Namely, we propose a special case of the problem for which any Branch and Bound algorithm with any polynomial time computed Lower Bound can't solve some instances even for n=60 operations in appropriate time. Since the best known algorithm [Sewell] solves all benchmark instances in less than 1 second per instances, we suggest for the future research not concentrating on exact algorithms which are tested only on benchmark instances, but concentrate on algorithms for special cases with a good theoretically estimated run time or on essentially new solution schemes.

Additionally, we propose some results for the maximization version of problem of SALBP-1 and present a technique to reduce the graph of precedence relations that provides some advantages.

The work was supported by 11-08-13121-офи-м-2011-РЖД.

References

Scholl A., Balancing and Sequencing of Assembly Lines, Physica Verlag, A Springer-Verlag Company, 1999, 318 p.

Sewell E.C., Jacobson S.H., A Branch, Bound, and Remember Algorithm for the Simple Assembly Line Balancing Problem, INFORMS Journal on Computing, doi 10.1287/ijoc.1110.0462, 2011, pp. 1--10.

Hackman S.T., Mgazine M.J., Wee T.S., Fast, Effective Algorithms for Simple Assembly Line Balancing Problems, Operations Research, N 37, 1989, 916 -- 924.

Finkelstein Yu. Yu., Approximate Methods and Applied Problems of Discrete Optimization, Nauka, Moscow, 1976 (in Russian).

Posypkin M.A., Sigal I. Kh., Speedup estimates for some variants of the parallel implementations of the branch-and-bound method, Computational Mathematics and Mathematical Physics, Vol. 46, N 12, 2006, 2189 --2 202.

Aloulou M.A. and Artigues C.,Flexible solutions in disjunctive scheduling: general formulation and study of the flow-shop case. Computers and Operations Research, 37(5), 2010, 890 -- 898.

Aloulou M.A., Kovalyov M.Y., Portmann M.-C., Maximization Problems in Single Machine Scheduling}, Annals of Operations Research, 129, 2004, 21 -- 32.

Arkin E.M., Bender M.A., Mitchell J.S.B., Skeinab S.S., The Lazy Bureaucrat Scheduling Problem, Information and Computation, 184, 2003, 129 – 146.

Gafarov E.R., Lazarev A.A., Werner F., Transforming a Pseudo-Polynomial Algorithm for the Single Machine Total Tardiness Maximization Problem into a Polynomial One, Annals of Perations Research, 2011 (in print).

Queyranne M., Bounds for Assembly Line Balancing Heuristics, Operations Research, 33(6), 1985, 1353-1359.

Lazarev A.A., Gafarov E.R.,Transformation of the Network Graph of Scheduling Problems with Precedence Constraints to a Planar Graph, Doklady Mathematics, 2009, Vol. 79, N 1, 1-3.