Computers & Operations Research 79 (2017) 34–38

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

A note on a single-machine lot scheduling problem with indivisible orders

Dar-Li Yang, Yung-Tsung Hou, Wen-Hung Kuo[*]

Department of Information Management, National Formosa University, Yunlin 632, Taiwan, ROC

ARTICLE INFO

Keywords:
Lot scheduling
Single machine
Total completion time
Indivisible order
Integer programming

ABSTRACT

In this paper, a lot scheduling problem on a single machine with indivisible orders is studied. The objective is to minimize the total completion time of all orders. We show that the problem is NP-hard in the strong sense. Then, a binary integer programming approach and four simple heuristics are proposed to solve the problem. The binary integer programming approach with running time limit is considered as one heuristic method. As compared to a lower bound, the average performances of the heuristic method are really good and better than those of the four simple heuristics.

1. Introduction

Generally, there are two main production processes in a production system, that is, continuous production and batch production. Here, we are interested in batch production. In the literature, there are two categories of batch scheduling problems. One is batch scheduling with divisible batch sizes. For example, Santos and Magazine studied a single machine scheduling problem with divisible lots. The objective is to minimize the total flowtime of all jobs. They showed that lot splitting may occur and that the batching decisions may have a great impact on the lead time of jobs. Naddef and Santos [1] studied a single machine problem with batching jobs. The objective is to minimize the total completion times. They showed that the greedy algorithm solves the problem if jobs are all of one type. They also provided a heuristic for the problem with various job types. Coffman et al. [2] considered a single machine job shop in which subassemblies of two different types are made and then assembled into products. They provided an efficient algorithm for minimizing the total flow time of the products. Dobson et al. [3] considered batch jobs in the multiple-machine scheduling problem. The objective is to minimize the mean flow times. They proposed an efficient algorithm for computing the optimal solution for single product case. Hou et al. [4] studied a lot scheduling problem with orders which can be split. Orders are grouped into lots and then processed. The objective is to minimize the total completion time of all orders. They showed that this problem can be solved in polynomial time.

The other is batch scheduling with indivisible batch sizes. Shallcross [5] studied a problem of batching identical jobs on a single machine. He presented an algorithm to minimize the sum over all jobs of the batched completion times. Mosheiov et al. [6] addressed a classical minimum flow-time, single-machine, batch-scheduling problem. They introduced a simple rounding procedure for Santos and Magazine's solution [7], which guarantees optimal integer batches. Mor and Mosheiov [8] studied an identical parallel-machine scheduling problem with identical job processing times and identical setups. They showed that the solution is given by a closed form, consisting of identical decreasing arithmetic sequences of batch sizes on the different machines. For more studies of this line, the reader is referred to the survey papers (Potts and Kovalyov [9], and Allahverdi et al. [10]).

In a factory, products are usually made according to customers' orders. This production approach is called MTO (make to order). Since different orders may contain different quantities, two production strategies are applied in the batch production, especially when the lot size of the batch production is fixed. Also, in this particular situation, the production time of each lot is fixed no matter how many quantities in the lot. Therefore, from the viewpoint of efficiency, one order may be divided into several lots to fill up each lot. The study presented by Hou et al. [4] is based on this viewpoint. However, from the viewpoint of management, one order is not divided into different production lots because the products of the same order are finished at the same time and then delivered to the customer. Based on this viewpoint, in this paper, we study the same problem given by Hou et al. [4] but orders are restricted to be indivisible.

2. Problem description

There are n orders (O_i, i=1, 2, ..., n) to be grouped into lots and then be processed on a single machine. Every order has its own size (σ_i, i=1, 2, ..., n). The size of each order is no more than one lot's capacity (k). On top of that, every order is indivisible. It means products of each

[*] Corresponding author.
E-mail address: whkuo@nfu.edu.tw (W.-H. Kuo).

http://dx.doi.org/10.1016/j.cor.2016.10.004
Received 13 October 2014; Received in revised form 21 April 2016; Accepted 9 October 2016
Available online 11 October 2016
0305-0548/ © 2016 Elsevier Ltd. All rights reserved.

individual order have to be processed on the same lot. The orders in the same lot have the same processing time (t). Therefore, all orders in the same lot have the same completion time.

The machine can handle at most one lot at a time and cannot stand idle until the last lot assigned to it has finished processing. The objective is to minimize the total completion time ($\sum C_{O_i}$) of all orders. Thus, using the three-field notation, this scheduling problem is denoted by $1/lot,\ indivisible/\sum C_{O_i}$.

3. The analysis of the $1/lot,\ indivisible/\sum C_{O_i}$ problem

The proposed problem is similar to the classical bin-packing problem when “lots” are considered as “bins” and “orders” as “items”. The only difference is that the objective function of the bin-packing problem is to find the minimum number of bins while that of the proposed problem is to minimize the total completion time of all orders. In other words, the contribution of each bin to the objective function in the bin-packing problem is the same. However, the importance of each lot is different in the proposed problem. The completion time of a latter lot is longer than that of an earlier one. Therefore, it is better to arrange orders in earlier lots to minimize the total completion time. Based on the above observation, a similar reduction to 3-partition problem is used to prove the strong NP-hardness of the problem $1/lot,\ indivisible/\sum C_{O_i}$.

Theorem 1. The problem $1/lot,\ indivisible/\sum C_{O_i}$ is NP-hard in the strong sense.

Proof. We show that 3-partition problem reduces to this problem. Considering the following well-known NP-complete problem:

3-partition: Given positive integers a_1, a_2, ..., a_{3m}, B and for each $j \in A = \{1, 2, \dots, 3m\}$ such that $B/4 < a_j < B/2$ and $\sum_{j\in A} a_j = mB$, does there exist disjoint sets A_1, A_2, ..., A_m of A such that $\sum_{j\in A_1} a_j = \sum_{j\in A_2} a_j = \dots = \sum_{j\in A_m} a_j = B$?

First, an instance of the $1/lot,\ indivisible/\sum C_{O_i}$ problem is constructed as follows.

$N = 3m,\ \ K = B,\ \ t = 1,\ \ \sigma_i = a_i$ where $i = 1, 2, \dots, 3m$.

We will show that 3-partition problem has a solution if and only if the above instance has an optimal schedule with minimal total completion time $\sum C_{O_i} = 3m(m+1)/2$.

($\Rightarrow$) If 3-partition problem has a solution, then there exist disjoint sets A_1, A_2, ..., A_m of A such that $\sum_{j\in A_1} a_j = \sum_{j\in A_2} a_j = \dots = \sum_{j\in A_m} a_j = B$. Let the orders corresponding to A_1 be assigned to J_1, those corresponding to A_2 be assigned to J_2, ... and so on. Since $B/4 < a_j < B/2$, exactly three orders are assigned to each job. Therefore, the total completion time of the above instance is $\sum C_{O_i} = 3m(m+1)/2$.

($\Leftarrow$) If 3-partition problem has no solution, we will show that the total completion time of any schedule for the above instance is greater than $3m(m+1)/2$. Assume that 3-partition problem has no solution, then at least one disjoint set of A, say A_k, in which $\sum_{j\in A_k} a_j < B$. Therefore, if the orders corresponding to A_k are assigned to one job, then the sum of a_j in the other disjoint sets is greater than $(mB - B)$. In such a situation, since $B/4 < a_j < B/2$, the best feasible schedule to minimize the total completion time is arranged as follows. The first $(m-1)$ jobs consist of three orders, the mth job consists of two orders and the (m+1)th job consists of one order. Hence, the total completion time is $\sum C_{O_i} = 3(1 + 2 + \dots + (m-1)) + 2m + (m+1) = (3m(m+1)/2) + 1$. Therefore, if 3-partition problem has no solution, the total completion time of the above instance is greater than $3m(m+1)/2$. This completes the proof.

4. Integer programming formulation

Let $X_{i[q]} = 1$ if the ith order is assigned to the qth lot, and 0 otherwise. Since the processing time of each lot is t, the completion times of the first lot, the second one, ... are t, $2t$, ..., respectively. Thus, the total completion time of all orders is $t\sum_{q=1}^{N}\sum_{i=1}^{N} X_{i[q]}q$. Then a binary integer programming (BIP) formulation to solve the proposed problem is developed as follows.

$$\text{Minimize}\quad t\sum_{q=1}^{N}\sum_{i=1}^{N} X_{i[q]}q \tag{1}$$

$$\text{Subject to}\quad \sum_{q=1}^{N} X_{i[q]} = 1 \quad i = 1, 2, \dots, N \tag{2}$$

$$\sum_{i=1}^{N} \sigma_i X_{i[q]} \le K \quad q = 1, 2, \dots, N \tag{3}$$

$$X_{i[q]} \in \{0, 1\} \quad i = 1, 2, \dots, N,\ \ q = 1, 2, \dots, N \tag{4}$$

The objective is to minimize the total completion time of all orders which is shown in objective function (1). Eq. (2) ensures that each order is only assigned to one lot. Constraint (3) limits the total sizes of orders that are assigned to the same lot to the lot capacity (K). Finally, constraint (4) guarantees that variable $X_{i[q]}$ is either 0 or 1.

5. Heuristics

Since the proposed problem is similar to the bin packing problem, the following simple commonly used heuristics for solving the bin packing problem are adopted to find the solution of the proposed problem. In the first two heuristics, the orders are randomly arranged in a list. On the other hand, in the last two heuristics, the orders are arranged in a non-decreasing order of their sizes in a list because such an order sequence is optimal for the same problem with divisible orders [4]. The four simple heuristics are given as follows.

First Fit Random (FFR) algorithm.

Step 1. Arrange the orders in a list randomly.
Step 2. Select the order at the head of the list and assign it to the first feasible lot with enough residual capacity for the order from the beginning. If the order doesn’t fit in any existing lot, place the order in a new lot.
Step 3. Repeat Step 2 until all orders are assigned to a lot.

Best Fit Random (BFR) algorithm.

Step 1. Arrange the orders in a list randomly.
Step 2. Select the order at the head of the list and assign it to the most feasible lot with the least enough residual capacity for the order by scanning all the existing lots. If the order doesn’t fit in any existing lot, place the order in a new lot.
Step 3. Repeat Step 2 until all orders are assigned to a lot.

First Fit Non-decreasing (FFN) algorithm.

Step 1. Arrange the orders in a non-decreasing order of their sizes in a list.
Step 2 and Step 3 are the same as those in FFR algorithm.

Best Fit Non-decreasing (BFN) algorithm.

Step 1. Arrange the orders in a non-decreasing order of their sizes in a list.
Step 2 and Step 3 are the same as those in BFR algorithm.

The time complexities of first-fit and best-fit are $O(n \log n)$. Therefore, it is obvious that the time complexities of the four algorithms are all $O(n \log n)$.

Table 1
Computational results of experiments (BIP).

	σ_i=1–5, k=15			σ_i=1–5, k=30			σ_i=1–10, k=15			σ_i=1–10, k=30		
	Error (%)			Error (%)			Error (%)			Error (%)		
n	Average	Maximal	Optimal no.	Average	Maximal	Optimal no.	Average	Maximal	Optimal no.	Average	Maximal	Optimal no.
20	0	0	30	0	0	30	0	0	30	0	0	30
30	0	0	30	0	0	30	1.05	9.23	26	0	0	30
40	0.38	3.44	26	0	0	30	1.30	10.08	25	0.15	4.46	29
50	1.08	3.04	13	0	0	30	1.31	10.86	24	0.13	3.93	29
60	1.38	2.77	7	0	0	30	1.34	7.52	23	0.28	3.19	27
70	1.59	2.37	2	0	0	30	1.06	10.24	25	0	0	30
80	1.23	2.38	6	0.56	1.71	18	1.49	8.01	22	0.09	2.63	29
90	0.91	2.54	10	0.57	1.44	15	1.30	7.27	23	0.18	3.15	28
100	1.47	2.06	1	0.86	1.32	3	2.46	6.55	17	0.08	2.47	29

σ_i: Order size, k: Lot capacity, n: Order number.

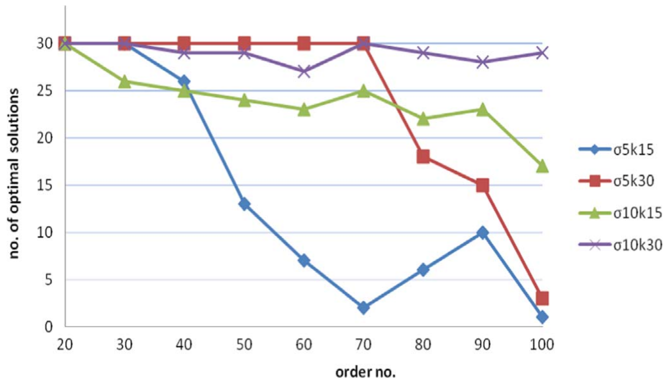

Fig. 1. The trend in the number of optimal solutions.

6. Computational experiments

The above binary integer programming approach can solve the proposed problem, but it is time-consuming when it comes to a large problem. Considering the efficiency of the BIP, the run time limit of the BIP is set to 3600 s. Also, in order to evaluate the performance of the BIP, it is compared to the above heuristics. They are all tested in the computational experiments which are conducted based on the following parameter set.

(1) Order number n is equal to 20, 30, 40, 50, 60, 70, 80, 90, 100.
(2) Lot capacity k is equal to 15, 30.
(3) Order size σ_i is uniformly distributed over [1,5], [1,10] ($\sigma_i \overset{d}{=} U(1,5)$, $\sigma_i \overset{d}{=} U(1,10)$)

There are $9 \times 2 \times 2 = 36$ problem types. For each problem type, 30 test problems are generated. Each test problem is solved by BIP, LP and four heuristics, respectively. BIP and LP are solved by using a computer program coded in LINGO 11.0 while the four heuristics are carried on a Matlab program with 4 GB of memory available for working storage, running on a personal computer Intel(R) Core(TM) i7-2600 CPU @3.4 GHz. To evaluate the performance of the computational results, we have to come up with a lower bound (LB) and then compare these percentage errors ($100*(H - LB)/LB$) in different test problems where H is BIP, or one of the four heuristics.

Obviously, one lower bound can be obtained from the solution of a variant of the original problem by changing the original problem to the one in which orders are divisible and can be processed in different lots. Therefore, we only need to change Eq. (4) as follows.

$$X_{i[q]} \geq 0 \quad i = 1, 2, \ldots, N, \;\; q = 1, 2, \ldots, N \qquad (4')$$

Then, since the problem becomes a linear programming (LP) problem, we take much less time to solve the problem than the original one. The lower bound is also tight because the solutions of the original problem and its variant can happen to be the same (integers).

The average and maximal percentage errors for the BIP solutions and the number of optimal solutions obtained within 3600 s in different test problems are shown in Table 1 and Fig. 1. The results of the four heuristics are also shown from Tables 2–5.

When comparing the results from Tables 1–5, we have the following observations:

(1) If the orders are arranged in a non-decreasing order of their sizes first, the results of first-fit and best-fit algorithms are the same and much better than those with random order sequences.

Table 2
Computational results of experiments (FFR).

	σ_i=1–5, k=15			σ_i=1–5, k=30			σ_i=1–10, k=15			σ_i=1–10, k=30		
	Error (%)			Error (%)			Error (%)			Error (%)		
n	Average	Maximal	Optimal no.	Average	Maximal	Optimal no.	Average	Maximal	Optimal no.	Average	Maximal	Optimal no.
20	22.71	36.61	0	16.53	29.03	0	27.64	37.93	0	22.69	48.81	0
30	24.82	41.16	0	18.97	33.33	0	27.63	36.86	0	24.36	36.03	0
40	28.33	48.15	0	23.31	28.27	0	28.63	38.18	0	27.52	38.62	0
50	27.50	41.03	0	24.61	38.78	0	27.87	39.84	0	29.45	42.22	0
60	28.55	38.48	0	27.14	32.93	0	29.00	41.97	0	29.31	43.70	0
70	29.07	39.72	0	29.68	38.44	0	29.55	39.43	0	33.56	45.30	0
80	31.22	44.89	0	28.93	36.40	0	30.62	39.21	0	32.84	42.17	0
90	30.48	37.94	0	30.00	40.73	0	30.56	49.10	0	34.14	43.62	0
100	30.81	39.18	0	29.14	36.30	0	31.68	41.87	0	33.97	46.79	0

σ_i: Order size, k: Lot capacity, n: Order number.

Table 3
Computational results of experiments (BFR).

	σ_i=1–5, k=15			σ_i=1–5, k=30			σ_i=1–10, k=15			σ_i=1–10, k=30		
	Error (%)			Error (%)			Error (%)			Error (%)		
n	Average	Maximal	Optimal no.	Average	Maximal	Optimal no.	Average	Maximal	Optimal no.	Average	Maximal	Optimal no.
20	22.34	36.61	0	16.30	29.03	0	25.47	37.93	0	23.78	48.81	0
30	24.82	41.16	0	18.97	33.33	0	27.35	36.86	0	24.41	36.03	0
40	28.33	48.15	0	23.31	28.27	0	28.21	38.18	0	27.64	38.62	0
50	27.50	41.03	0	24.61	38.78	0	27.66	35.70	0	29.67	42.22	0
60	28.55	38.48	0	27.14	32.93	0	28.99	41.97	0	29.49	43.70	0
70	29.07	39.72	0	29.68	38.44	0	29.58	39.43	0	33.78	45.30	0
80	31.22	44.89	0	28.93	36.40	0	30.55	38.82	0	32.82	42.17	0
90	30.48	37.94	0	30.00	40.73	0	30.40	49.10	0	34.01	43.62	0
100	30.81	39.18	0	29.14	36.30	0	31.43	41.87	0	34.04	46.79	0

σ_i: Order size, k: Lot capacity, n: Order number.

Table 4
Computational results of experiments (FFN).

	σ_i=1–5, k=15			σ_i=1–5, k=30			σ_i=1–10, k=15			σ_i=1–10, k=30		
	Error (%)			Error (%)			Error (%)			Error (%)		
n	Average	Maximal	Optimal no.	Average	Maximal	Optimal no.	Average	Maximal	Optimal no.	Average	Maximal	Optimal no.
20	8.37	12.76	0	6.53	10.52	0	23.63	37.70	0	10.19	15.36	0
30	7.73	11.13	0	5.16	8.53	0	20.99	31.95	0	7.99	12.25	0
40	6.54	11.54	0	3.85	6.00	0	23.81	37.49	0	8.43	12.28	0
50	7.34	11.38	0	3.80	6.62	0	23.99	35.89	0	7.89	11.27	0
60	7.67	10.07	0	3.18	6.85	0	23.61	32.43	0	7.80	12.26	0
70	7.33	10.92	0	3.08	5.07	0	23.34	31.03	0	7.28	11.11	0
80	7.32	10.23	0	2.94	5.12	0	22.99	31.46	0	7.28	9.68	0
90	6.87	9.55	0	3.02	4.82	0	21.79	31.22	0	7.00	10.36	0
100	7.15	9.30	0	2.71	3.96	0	22.91	30.89	0	6.74	8.95	0

σ_i: Order size, k: Lot capacity, n: Order number.

Table 5
Computational results of experiments (BFN).

	σ_i=1–5, k=15			σ_i=1–5, k=30			σ_i=1–10, k=15			σ_i=1–10, k=30		
	Error (%)			Error (%)			Error (%)			Error (%)		
n	Average	Maximal	Optimal no.	Average	Maximal	Optimal no.	Average	Maximal	Optimal no.	Average	Maximal	Optimal no.
20	8.37	12.76	0	6.53	10.52	0	23.63	37.70	0	10.19	15.36	0
30	7.73	11.13	0	5.16	8.53	0	0	7.89	11.27	0		
60	7.67	10.07	0	3.18	6.85	0	23.61	32.43	0	7.80	12.26	0
70	7.33	10.92	0	3.08	5.07	0	23.34	31.03	0	7.28	11.11	0
80	7.32	10.23	0	2.94	5.12	0	22.99	31.46	0	7.28	9.68	0
90	6.87	9.55	0	3.02	4.82	0	21.79	31.22	0	7.00	10.36	0
100	7.15	9.30	0	2.71	3.96	0	22.91	30.89	0	6.74	8.95	0

σ_i: Order size, k: Lot capacity, n: Order umber.

(2) Even though the performances of FFN and BFN are better than those of the other two heuristics, only in some test problems of the problem type (σ_i=1–5, k=30) the average percentage errors are less than 5. The average percentage errors are even more than 20 in the problem type (σ_i=1–10, k=15). Therefore, the performances of FFN and BFN are not good, let alone those of FFR and BFR. In addition, no optimal solution is found in all test problems. The four heuristics are commonly used to solve the bin packing problem. They are supposed to have good performance in the similar proposed problem. However, they don't. The reason may be the objective function of the proposed problem is different from that of the bin packing problem.

(3) Based on Observation (2), although BIP takes much time to find the solutions, the performances of BIP with time limit are much better than those of the four heuristics. Therefore, it can be regarded as a good heuristic method.

Next, we further discuss the results of BIP with running time limit. From Table 1 and Fig. 1, we have the following observations:

(1) For n = 20, the optimal solutions for all generated test problems can be found within 3600 s.
(2) The larger the lot capacity is or the smaller the order size range is, the more optimal solutions you can obtain within the running time

limit.

(3) There exists one trend, that is, the number of optimal solutions decreases when the number of orders increases.
(4) All average percentage errors are less than 2.5, it means that the performance of the BIP with running time limit is really good, especially, in the problem type with parameters k=30 and $\sigma_i \overset{d}{=} U(1,5)$.
(5) Most maximal percentage errors are less than 4.5, it implies that the BIP performs well in most test problems, even in the worst situations. However, some of them in the problem type with parameters k=15 and $\sigma_i \overset{d}{=} U(1,10)$ are greater than 10, though their average percentage errors are less than 2.5. The performance of BIP in such a problem type is not robust. Therefore, it is worthwhile to come up with other heuristics with better and robust performances.

7. Conclusion

In this paper, we study a single-machine lot scheduling problem with indivisible orders. First, the problem is proved to be NP-hard in the strong sense. Next, a binary integer programming approach and four heuristics are proposed to solve the problem. Considering the efficiency of the BIP, the run time limit is set. As compared to the lower bound, it turns out the average performances of the BIP with running time limit are really good for all test problems. The performances of the BIP with running time limit are also much better than those of the four heuristics. The maximal percentage errors of the BIP with running time limit are a little greater than 10 in one situation. Therefore, it is worthwhile to find other heuristics with better performance in the future.

Acknowledgement

This research was supported in part by the National Science Council, Taiwan, R.O.C. under Grant number NSC-102–2221-E-150-043-MY2 and the Ministry of Science and Technology, Taiwan, R.O.C. under Grant number MOST-102-2221-E-105-025.

References

[1] Naddef D, Santos C. One-pass batching algorithms for the one-machine problem. Discret Appl Math 1988;21:133–45.
[2] Coffman ED, Nozari A, Yannakakis M. Optimal scheduling of products with two subassemblies on single machine. Oper Res 1989;37:426–36.
[3] Dobson G, Karmarkar UD, Rummel JL. Batching to minimize flow times on parallel heterogeneous machines. Manag Sci 1989;35:607–13.
[4] Hou YT, Yang DL, Kuo WH. Lot scheduling on a single machine. Inf Process Lett 2014;114:718–22.
[5] Shallcross DF. A polynomial algorithm for a one machine batching problem. Oper Res Lett 1992;11:213–8.
[6] Mosheiov G, Oron D, Ritov Y. Minimizing flow-time on a single machine with integer batch sizes. Oper Res Lett 2005;33:497–501.
[7] Santos C, Magazine M. Batching in single operation manufacturing systems. Oper Res Lett 1985;4:99–103.
[8] Mor B, Mosheiov G. Batch scheduling of identical jobs on parallel identical machines. Inf Process Lett 2012;112:762–6.
[9] Potts CN, Kovalyov MY. Scheduling with batching: a review. Eur J Oper Res 2000;120:228–49.
[10] Allahverdi A, Ng CT, Cheng TCE, Kovalyov MY. A survey of scheduling problems with setup times or costs. Eur J Oper Res 2008;187:985–1032.