SINGLE MACHINE SCHEDULING WITH BATCH DELIVERY TO MULTIPLE CUSTOMERS IN A STAR-SHAPED NETWORK

LEIYANG WANG
Department of Mathematics
East China University of Science and Technology
Shanghai 200237, P. R. China
wangleiyang1983@163.com

ZHAOUI LIU*
Department of Mathematics
East China University of Science and Technology
Shanghai 200237, P. R. China
zhliu@ecust.edu.cn

Published 23 January 2013

In this paper, we consider the scheduling problem in which the jobs are first processed on a single machine and then delivered in batches by a single vehicle with limited capacity to the respective customers located at the vertices of a star-shaped network. The goal is to minimize the makespan. We present a 3/2-approximation algorithm for the identical job size case and a 2-approximation algorithm for the non-identical job sizes case.

Keywords: Scheduling; batch delivery; star-shaped network; approximation algorithm.

1. Introduction

In this paper, we study the scheduling problem in which the jobs are first processed by a single machine and then delivered by a single vehicle with limited capacity to the respective customers located at the vertices of a star-shaped network. The problem can be defined as follows. Let \( S = (V \cup \{O\}, E) \) be an undirected star-shaped network, where \( V = \{1, 2, \ldots, n\} \) is a set of \( n \) vertices, \( O \) is a designated vertex called center, and \( E \) is the set of edges. The network consists of internally disjoint \( h \) paths \( S_1, S_2, \ldots, S_h \) with the common endpoint \( O \), and the vertices in \( V \) are distributed on the paths (see Fig. 1). We call each \( S_i \) a branch, for \( i = 1, 2, \ldots, h \). Let \( N = \{J_1, J_2, \ldots, J_n\} \) be a set of \( n \) jobs to be processed by a single machine located at the center \( O \). For each \( j \in \{1, 2, \ldots, n\} \), the processing time of job \( J_j \) is \( p_j \) (\( p_j \geq 0 \)), and job \( J_j \) must be delivered to customer \( j \) located at vertex \( j \) after...
L. Wang & Z. Liu

Fig. 1. Star-shaped network.

being processed. A vehicle is available to deliver the jobs. Job \( J_j \) has size \( s_j \), which represents the physical space \( J_j \) occupies when it is delivered by the vehicle. The capacity of the vehicle, which represents the total physical space that the vehicle provides for each delivery, is \( z \). Jobs delivered together compose a delivery batch. For job \( J_j \), its delivery time \( t_j \) is given by the two-way travel time between the center \( O \) and the vertex \( j \). All jobs and the single vehicle are available at the center at time zero. The goal is to determine the processing sequence of the jobs on the single machine and the delivery rule, so as to minimize the makespan, i.e., the time by which the vehicle has delivered the last job and returned to the center.

Motivated by applications in manufacturing and logistics, the above problem and its variants have absorbed attention of some researchers in recent years (see Cheng et al., 1996, 1997; Lee and Chen, 2001; Chang and Lee, 2004; Chen and Vairaktarakis, 2005; Hall and Potts, 2005; Li et al., 2005; Zhong et al., 2007; Geismar et al., 2008, etc.). If the customers locate at the same site, and the jobs have the same size, a polynomial optimization algorithm has been presented in Ahmadi et al. (1992) and Lee and Chen (2001). However, the problems in which the customers are located on a general network or the jobs have different sizes are NP-hard in the strong sense since the former includes the classical Traveling Salesman Problem (TSP) as a special case with \( p_j = 0 \) (\( j = 1, 2, \ldots, n \)), and the latter includes the Bin Packing Problem as a special case.

Li et al. (2005) studied the identical job size problem where the objective is to minimize the sum of job arrival times and provided a polynomial time algorithm for the case where the number of customers is constant. Levin and Penn (2008) considered the similar problem with an arbitrary number of customers and a single uncapacitated delivery vehicle, and presented a \( 6e \approx 16.309691 \)-approximation algorithm.

If the jobs have different sizes, Chang and Lee (2004) presented a \( 5/3 \)-approximation algorithm for the case in which the customers locate at the same site, and a \( 2 \)-approximation algorithm for the case in which the customers locate at two different sites. Later, Zhong et al. (2007) presented an improved algorithm with the
Single Machine Scheduling with Batch Delivery to Multiple Customers

worst-case performance ratio arbitrarily close to 3/2 for the case in which the customers locate at the same site. However, the algorithm requires a fully polynomial time approximation scheme (FPTAS) for the knapsack problem. Chang and Lee (2004), Zhong et al. (2007) and Su et al. (2009) also investigated the related two-machine problem where the customers locate at the same site.

In this paper, we consider the single-machine problem in which the customers locate at the vertices of a star-shaped network. The remainder of the paper is organized as follows. In Sec. 2, we introduce some preliminary results. Section 3 contains a 3/2-approximation algorithm for the identical job size case. Finally, a 2-approximation algorithm for the non-identical job sizes case is provided in Sec. 4.

2. Preliminary

We begin this section by considering the complexity of the problem. Note that the non-identical job sizes case is NP-hard in the strong sense as explained in Sec. 1. For the identical job size case, consider the star-shaped network with only one branch, i.e., a path with the machine located at some endpoint. Then, the delivery time of each batch equals to the maximum delivery time of the jobs in the batch. Thus, viewing the vehicle as a batch machine that can handle up to \( z \) jobs simultaneously, the identical job size case is equivalent to the two-machine flow shop problem with a batch machine at the second stage, which has been proved NP-hard in Potts et al. (2001).

The following lemma allows us to consider only some special schedules.

**Lemma 1.** There exists an optimal solution such that each delivery batch contains only jobs with customers on the same branch.

**Proof.** Since the customers are located on a star-shaped network, the vehicle must pass by the center when it travels between two customers on different branches. Then a delivery batch containing jobs with customers on different branches can be repartitioned into several batches, each of which contains only jobs with customers on the same branch. This does not increase the makespan.

Let \( B_i \) be a delivery batch produced by some algorithm. We denote the total processing time of the jobs in batch \( B_i \) by \( P_i \), and the delivery time of batch \( B_i \) by \( T_i \). By Lemma 1, we can assume that the customers associated with the jobs in \( B_i \) locate on the same branch. Then, \( T_i \) is equal to the maximum delivery time of the jobs in \( B_i \). If all delivery batches have been formed, our problem reduces to the classical two-machine flow shop problem (denoted by \( F2 \)) where each delivery batch \( B_i \) is viewed as a job with processing time \( P_i \) on machine 1 and processing time \( T_i \) on machine 2. Johnson (1954) provided a rule (named as Johnson’s rule) to solve \( F2 \): Scheduling the batches with \( P_i \leq T_i \) first in non-decreasing order of \( P_i \), and then scheduling the batches with \( P_i > T_i \) in non-increasing order of \( T_i \).
3. Identical Job Size

In this section, we present an approximation algorithm for the identical job size case. For convenience, we assume \( s_j = 1 \), for \( j = 1, 2, \ldots, n \). Thus, the capacity \( z \) represents the maximum number of jobs that can be delivered at the same time.

The algorithm constructs the delivery batches mostly according to the FB-LDT rule: Consider the jobs in non-increasing order of their delivery times (LDT), and partition them into batches by the Full Batch (FB) rule, i.e., every \( z \) successive jobs compose a batch. Note that according to the FB rule, every batch is full except the last one (if the number of jobs is not a multiple of \( z \)).

For each branch, first apply the FB-LDT rule to the jobs with customers on the branch. Let \( T_{FB-LDT}^j \) and \( T_{max} \) be the total delivery time and the maximum delivery time of the batches formed by the FB-LDT rule, respectively. Let \( B_{max} \) be a batch with delivery time \( T_{max} \). If \( T_{max} > \frac{1}{2} T_{FB-LDT}^j \), we will repartition the jobs in \( B_{max} \) by a dynamic programming.

Suppose that \( B_{max} = \{J_1, J_2, \ldots, J_m\} \) satisfying \( t_1 \geq t_2 \geq \cdots \geq t_m \), where \( m \leq z \). Consider the subproblem with only the jobs in \( B_{max} \). Obviously, the jobs in the subproblem should be processed in LDT order since their customers locate at the same branch and \( m \leq z \). Define \( F(i) \) as the minimum makespan to schedule \( J_1, J_2, \ldots, J_i \), where \( i = 1, 2, \ldots, m \). Define \( P(i) = \sum_{j=1}^{i} p_j \). If the last batch in an optimal subschedule for \( J_1, J_2, \ldots, J_i \) consists of \( J_h, J_{h+1}, \ldots, J_i \), then the batch should be delivered at time \( \max \{P(i), F(h-1)\} \). Thus, we have the following recurrence relation:

\[
F(i) = \min_{1 \leq h \leq i} \{\max\{P(i), F(h-1)\} + t_h\},
\]

for \( i = 1, 2, \ldots, m \). The boundary condition is \( F(0) = 0 \). The dynamic programming solves the subproblem in \( O(z^2) \) time. The optimal value \( F \) of the subproblem is a lower bound of the optimal makespan \( C^* \) of our problem. Let \( P = \sum_{j=1}^{n} p_j \). Then, the following lemma holds.

**Lemma 2.** \( C^* \geq \max\{P, T_{FB-LDT}^j, F\} \).

Now we present our approximation algorithm \( H1 \).

**Algorithm \( H1 \)**

Step 1. For each branch \( S_i \), \( i = 1, 2, \ldots, h \), apply the FB-LDT rule to the jobs with customers on \( S_i \). Let \( b_1 \) denote the total number of resulting batches.

Step 2. If \( T_{max} \leq \frac{1}{2} T_{FB-LDT}^j \), then apply Johnson’s rule to schedule the batches and stop, where the jobs in the same batch are sequenced in any order, else go to Step 3.

Step 3. First apply the above dynamic programming to schedule the jobs in \( B_{max} \).

Then, schedule the other batches in any order after the jobs in \( B_{max} \).

**Lemma 3.** If \( T_{max} \leq \frac{1}{2} T_{FB-LDT}^j \), then \( C_{max}^{H1} / C_{max} \leq \frac{3}{2} \).
**Proof.** Suppose that the batches formed by the FB-LDT rule are scheduled as \( B_1, B_2, \ldots, B_k \) by Johnson’s rule. Let \( k \) be the smallest index so that the vehicle operates continuously from delivering the batch \( B_k \). Then, \( C_{\text{max}}^{H_1} = \sum_{i=1}^{k} P_i + \sum_{i=k}^{b_1} T_i \).

**Case 1:** \( P_k \leq T_k \). In this case, we have \( P_i \leq T_i \) for \( i = 1, 2, \ldots, k \), by Johnson’s rule. Thus,

\[
C_{\text{max}}^{H_1} \leq \sum_{i=1}^{k} T_i + \sum_{i=k}^{b_1} T_i = T_k + T^{FB-LDT} \leq \frac{3}{2} T^{FB-LDT} \leq \frac{3}{2} C_{\text{max}}^*.
\]

**Case 2:** \( P_k > T_k \). In this case, it holds that \( P_i > T_i \) for \( i = k, k+1, \ldots, b_1 \), by Johnson’s rule. Thus,

\[
C_{\text{max}}^{H_1} \leq \sum_{i=1}^{k} P_i + T_k + \sum_{i=k+1}^{b_1} P_i = T_k + P \leq \frac{1}{2} T^{FB-LDT} + C_{\text{max}}^* \leq \frac{3}{2} C_{\text{max}}^*.
\]

**Lemma 4.** If \( T_{\text{max}} > \frac{1}{2} T^{FB-LDT} \), then \( C_{\text{max}}^{H_1} / C_{\text{max}}^* \leq \frac{3}{2} \).

**Proof.** Note that \( F \) is the optimal value to schedule the jobs in \( B_{\text{max}} \). If the vehicle operates continuously after time \( F \), then we have

\[
C_{\text{max}}^{H_1} = F + (T^{FB-LDT} - T_{\text{max}}) < F + \frac{1}{2} T^{FB-LDT} \leq \frac{3}{2} C_{\text{max}}^*.
\]

Otherwise, we have

\[
C_{\text{max}}^{H_1} < P + (T^{FB-LDT} - T_{\text{max}}) < P + \frac{1}{2} T^{FB-LDT} \leq \frac{3}{2} C_{\text{max}}^*.
\]

**Theorem 1.** \( C_{\text{max}}^{H_1} / C_{\text{max}}^* \leq \frac{3}{2} \) and the bound is tight.

**Proof.** It is obvious that \( C_{\text{max}}^{H_1} / C_{\text{max}}^* \leq \frac{3}{2} \) from Lemmas 3 and 4. To illustrate the bound is tight, we consider an instance with four customers located on the same branch. Let \( z = 2 \), and

\[
\begin{align*}
p_1 &= \frac{1}{2}, & t_1 &= 1 + 2\epsilon, \\
p_2 &= \frac{1}{2}, & t_2 &= 1 + \epsilon, \\
p_3 &= \epsilon, & t_3 &= 1, \\
p_4 &= 1 - \epsilon, & t_4 &= \epsilon,
\end{align*}
\]
For each branch

Step 1.

Given job list, the FF rule places the current job on the branch in a list in LDT order, and then apply the classical First Fit (FF) rule that solves the Bin Packing Problem, to divide the jobs into batches. For a LDT branch, our algorithm uses the FF-LDT rule to construct the delivery batches: First place the jobs with customers on the branch in a list in LDT order, and then apply the classical First Fit (FF) rule to the jobs

Thus, \( C_{\text{max}}^{H1} = 3 + 2\epsilon \). But according to the optimal solution, we have three batches \( B_1^* = \{ J_3 \} \), \( B_2^* = \{ J_1, J_2 \} \), \( B_3^* = \{ J_4 \} \), and \( C_{\text{max}}^* = 2 + 4\epsilon \). Hence, \( C_{\text{max}}^{H1}/C_{\text{max}}^* \to \frac{3}{2} \) as \( \epsilon \to 0 \).

4. Non-Identical Job Sizes

In this section, we consider the non-identical job sizes case. Note that it includes the Bin Packing Problem as a special case. For each branch, our algorithm uses the FF-LDT rule to construct the delivery batches: First place the jobs with customers on the branch in a list in LDT order, and then apply the classical First Fit (FF) rule that solves the Bin Packing Problem, to divide the jobs into batches. For a given job list, the FF rule places the current job \( J_j \) into the first bin (batch), where it fits; if such a bin does not exist, then places \( J_j \) into a new bin. Note that the FF rule is based on the job sizes (\( s_j \)) and the vehicle capacity (\( z \)).

Algorithm \( H2 \)

Step 1. For each branch \( S_i \), \( i = 1, 2, \ldots, h \), apply the FF-LDT rule to the jobs with customers on \( S_i \). Let \( b_2 \) be the total number of resulting batches.

Step 2. Apply Johnson’s rule to schedule the batches, where the jobs in the same batch are sequenced in any order.

For each branch \( S_i \), \( i = 1, 2, \ldots, h \), define \( T(i) \) (respectively \( T^*(i) \)) as the total delivery time of the batches formed by the FF-LDT rule (respectively in some optimal schedule) for the jobs with customers on \( S_i \). Then, we have the following lemma.

 Lemma 5. \( T(i)/T^*(i) \leq \frac{12}{7} \) for \( i = 1, 2, \ldots, h \).

Proof. Let \( J_{i_1}, J_{i_2}, \ldots, J_{i_m} \) be the jobs with customers on \( S_i \), where \( t_{i_1} \geq t_{i_2} \geq \cdots \geq t_{i_m} \). Define \( V_{i,j} = \{ J_{i_1}, J_{i_2}, \ldots, J_{i_j} \} \) for \( j = 1, 2, \ldots, m \). Let \( b(V_{i,j}) \) (respectively \( b^*(V_{i,j}) \)) denote the number of batches obtained by the FF-LDT rule (respectively by some optimal rule) for the jobs in \( V_{i,j} \). Xia and Tan (2010) proved that the FF rule produces a 12/7-approximation solution for the Bin Packing Problem. Then, \( b(V_{i,j}) \leq \frac{12}{7} b^*(V_{i,j}) \). The vehicle must travel the edge \((i_j, i_{j+1})\) when delivering any batch containing some jobs in \( V_{i,j} \), so it travels \((i_j, i_{j+1})\) exactly \( b(V_{i,j}) \) times by algorithm \( H2 \) and at least \( b^*(V_{i,j}) \) times by some optimal rule. Hence,

\[
T(i) = \sum_{j=1}^{m} (t_{i_j} - t_{i_{j+1}}) b(V_{i,j}) \leq \frac{12}{7} \sum_{j=1}^{m} (t_{i_j} - t_{i_{j+1}}) b^*(V_{i,j}) \leq \frac{12}{7} T^*(i),
\]

where \( t_{i_{m+1}} = 0 \).
Remark. Zhang et al. (2001) considered a scheduling problem of minimizing the makespan on a single batch machine with non-identical job sizes, which is equivalent to the special case of our problem with one branch and \( p_j = 0 \) for all \( j \), and gave a 7/4-approximation algorithm. Lemma 5 improves their result and proof.

Let \( T_{FF-LDT} \) be the total delivery time by the FF-LDT rule, and \( T^* \) the total delivery time by some optimal rule.

**Corollary 1.** \( T_{FF-LDT} / T^* \leq \frac{12}{7} \).

**Proof.** Since \( T_{FF-LDT} = \sum_{i=1}^{h} T(i) \) and \( T^* = \sum_{i=1}^{h} T^*(i) \), according to Lemma 1, we have

\[
T_{FF-LDT} = \sum_{i=1}^{h} T(i) \leq \sum_{i=1}^{h} \frac{12}{7} T^*(i) = \frac{12}{7} T^*.
\]

Let \( B_{max} \) and \( B_{sec} \) be the batches formed by the FF-LDT rule with the maximum and second maximum delivery times. Let \( T_{max} \) and \( T_{sec} \) be the delivery times of \( B_{max} \) and \( B_{sec} \), respectively. Then, we have the following lemma.

**Lemma 6.** If \( b_2 \geq 2 \), then \( C^*_\max \geq T_{max} + T_{sec} \).

**Proof.** If \( b_2 \geq 2 \), then there are at least two batches in an optimal solution. If \( B_{max} \) and \( B_{sec} \) are delivered to customers located at different branches, then according to Lemma 1, \( C^*_\max \geq T_{max} + T_{sec} \) holds. If \( B_{max} \) and \( B_{sec} \) are delivered to customers at the same branch, then according to the FF-LDT rule, the jobs that are assigned to \( B_{max} \) before the second bin (batch \( B_{sec} \)) is opened, and the first job that is assigned to \( B_{sec} \), all have delivery times no less than \( T_{sec} \). In addition, all these jobs cannot be assigned to the same batch, so the second maximum batch delivery time in an optimal schedule is at least \( T_{sec} \), i.e., \( C^*_\max \geq T_{max} + T_{sec} \).

**Theorem 2.** \( \frac{C^{H2}_{\max}}{C^*_\max} \leq 2 \).

**Proof.** Suppose that the batches formed by the FF-LDT rule are scheduled as \( B_1, B_2, \ldots, B_{b_2} \) by Johnson’s rule. Let \( k \) be the smallest index so that the vehicle operates continuously from delivering the batch \( B_k \). Then, \( C^{H2}_{\max} = \sum_{i=1}^{k} P_i + \sum_{i=k}^{b_2} T_i \). If \( b_2 = 1 \), then the conclusion holds obviously. So we assume that \( b_2 \geq 2 \).

Note that \( C^*_\max \geq \max\{P, T_{max} + T_{sec}\} \). If \( k \geq b_2 - 1 \), then

\[
C^{H2}_{\max} = \sum_{i=1}^{k} P_i + \sum_{i=k}^{b_2} T_i \leq P + T_{b_2-1} + T_{b_2} \leq 2C^*_\max.
\]

In the following, we assume that \( k \leq b_2 - 2 \). We distinguish two cases.
L. Wang & Z. Liu

Case 1: $P_{k+2} > T_{k+2}$. In this case, we have $P_i > T_i$ for $i = k+2, k+3, \ldots, b_2$, by Johnson’s rule. Thus,
\[
C_{H2}^{max} = \sum_{i=1}^{k} P_i + T_k + T_{k+1} + \sum_{i=k+2}^{b_2} T_i \\
\leq P - P_{k+1} + T_k + T_{k+1} < 2C^*_\text{max}.
\]

Case 2: $P_{k+2} \leq T_{k+2}$. In this case, it holds that $P_i \leq T_i$ for $i = 1, 2, \ldots, k$, and $P_k \leq P_{k+1} \leq P_{k+2}$ by Johnson’s rule.

If $P_{k+3} > T_{k+3}$, then it holds that $P_i > T_i$ for $i = k+3, k+4, \ldots, b_2$, by Johnson’s rule. Since $C^*_\text{max} \geq T_{\text{max}} + T_{\text{sec}}$, we have $\frac{4}{3} C^*_\text{max} \geq T_k + T_{k+1} + T_{k+2}$.

Noticing that $T_{\text{FF-LDT}} \leq \frac{12}{14} C^*_\text{max}$, we have
\[
C_{H2}^{max} = \sum_{i=1}^{k} P_i + T_k + T_{k+1} + T_{k+2} + \sum_{i=k+3}^{b_2} T_i \\
\leq \frac{1}{3} \left( P - P_{k+1} - P_{k+2} + T_k + T_{k+1} + T_{k+2} \right) + \frac{2}{3} \left( P_k + T_{\text{FF-LDT}} \right) \\
\leq \frac{1}{3} \left( C^*_\text{max} - 2P_k + \frac{3}{2} C^*_\text{max} \right) + \frac{2}{3} \left( P_k + \frac{12}{7} C^*_\text{max} \right) \\
< 2C^*_\text{max}.
\]

If $P_{k+3} \leq T_{k+3}$, then it holds $P_k \leq P_{k+1} \leq P_{k+2} \leq P_{k+3}$ by Johnson’s rule. Thus, $C^*_\text{max} \geq P_k + P_{k+1} + P_{k+2} + P_{k+3} \geq 4P_k$. Hence
\[
C_{H2}^{max} = \sum_{i=1}^{k-1} P_i + P_k + \sum_{i=k}^{b_2} T_i \\
\leq P_k + T_{\text{FF-LDT}} \leq \frac{1}{4} C^*_\text{max} + \frac{12}{7} C^*_\text{max} < 2C^*_\text{max}.
\]

This completes the proof. □

Acknowledgments

The authors are grateful to the anonymous referees for their many helpful comments and suggestions. This research is supported by the National Natural Science Foundation of China under grant number 11171106 and the Fundamental Research Funds for the Central Universities of China.

References

Leiyang Wang is a PhD student at the Department of Mathematics, East China University of Science and Technology. His research interest is in scheduling, and he has made progress in the design of approximation algorithms for some scheduling problems with batch delivery consideration.

Zhaohui Liu is a Professor of Mathematics at East China University of Science and Technology (ECUST), Shanghai China. He received his BS degree in Operations Research from Fudan University in 1992, and PhD in Applied Mathematics from ECUST in 1998. His principal research interests are in scheduling and combinatorial optimization, and he has published a number of papers in Operations Research Letters, European Journal of Operational Research, IIE Transactions, Journal of Combinatorial Optimization, Networks, and other academic journals.