Scheduling of two-machine flowshop with outsourcing lead-time

Eun-Seok Kim[a] and Ik Sun Lee[b,*]

[a] *School of Business and Management, Queen Mary University of London, United Kingdom*

[b] *School of Business, Dong-A University, Seo-gu, Busan 602-760, Republic of Korea*

[*] Corresponding author. Tel.: +82-51-200-7425

E-mail addresses: e.kim@qmul.ac.uk (Eun-Seok Kim), lis1007@dau.ac.kr. (Ik Sun Lee)

Abstract

This paper considers a two-machine flowshop scheduling problem with outsourcing lead-time. While the first operation of jobs can be either processed in-house or outsourced to subcontractors, the second operation of jobs must be processed in-house. Outsourcing a job incurs a job-dependent outsourcing cost as well as outsourcing lead-time, which implies that the outsourced job becomes available for the second operation after its outsourcing lead-time. The objective is to minimize the weighted sum of the outsourcing costs and the scheduling costs which is either makespan or total completion time. The problems are shown to be NP-hard, and both exact and heuristic algorithms are developed for the problems. Finally, computational experiments show that the proposed algorithms provide efficient and effective solutions.

Keywords: Scheduling, Outsourcing, Two-machine flowshop, Branch-and-bound, Heuristic

1. Introduction

Outsourcing is the business practice of hiring a party outside a company to perform services and create goods that traditionally were performed in-house by the company's own resources. Since outsourcing was first recognized as a business practice in 1989 (Drucker, 1989), it is playing a more and more important role in recent manufacturing industries - proper outsourcing can shorten lead time, reduce total costs, and make an organization more flexible (Lee and Choi, 2011).

Outsourcing opportunities may exist in different forms in general (Qi, 2011). Some subcontractors may be able to provide services for all operations; some subcontractors may be dedicated to one operation only. The manufacturer may fully outsource all jobs to a subcontractor for one operation, or partially outsource a subset of the jobs. There are various variants of the problems to be studied. However, the practice of outsourcing makes the problems more complicated than the literature for various reasons. One reason is the transportation between the subcontractor and the manufacturer. In many cases, the subcontractors have their own production schedules in a different location from the manufacturer, which implies that it is unlikely the outsourced jobs to be delivered on time the manufacturer wants. For this reason, it is important to consider the outsourcing lead times.

This paper considers a two-machine flowshop model with jobs consisting of two operations. While the first operation of the jobs can be either processed in the first in-house machine or outsourced to a subcontractor, the second operation must be processed in the second in-house machine. If the first operation of a job is outsourced to a subcontractor, then the half-finished job is delivered from the subcontractor and completed in the second in-house machine. Outsourcing a job incurs a job-dependent outsourcing cost as well as outsourcing lead-time which reflects necessary production and delivery time from the subcontractor, and therefore the outsourced job becomes available for the second operation after its outsourcing lead-time. The objective is to minimize the

weighted sum of the outsourcing costs and the scheduling costs which is either makespan, that is, the latest completion time of the second operations of the jobs or total completion time.

Many studies have considered the outsourcing strategy in the various production and scheduling research area. Lee and Sung (2008a, b) studied single-machine scheduling to minimize the weighted sum of various scheduling measures and the outsourcing costs within an outsourcing budget constraint, where the scheduling measure is either one of the total completion time, the maximum lateness and the total tardiness. Zhong and Huo (2013) studied a single machine scheduling problem to minimize the weighted sum of total processing cost and either makespan or the number of tardy jobs, where there exists a delivery time as a stepwise function of outsourcing time for an outsourced job. Hong and Lee (2016) considered another single machine scheduling problem with subcontractor selection, in order to minimize the total outsourcing cost by considering the due dates constraints for each job and capacity limits of the subcontractors. Recently, Lu et al. (2020) considered single machine scheduling problems with outsourcing under different fill rates or quantity discount rates.

In the parallel machine environments, Chen and Li (2008) studied identical parallel scheduling to minimize the outsourcing costs within the makespan upper-limit. Mokhtari and Abadi (2013) considered parallel scheduling to minimize the weighted sum of total completion time and total outsourcing costs, where the in-house facility has some unrelated parallel machines while each subcontractor owns one single machine. Neto, Filho, and da Silva (2015) studied identical parallel machine scheduling to minimize the sum of total outsourcing costs and the total weighted tardiness within an outsourcing budget constraint. Liu, Lee, and Wang (2016) addressed identical parallel machine scheduling to minimize the sum of the total resource consumption and the total outsourcing costs within the upper-limit for the maximum tardiness of the jobs. Liao et al. (2020) studied a parallel-machine group scheduling problem where non-identical jobs with arbitrary sizes and inclusive processing set restrictions can be either processed on in-house parallel machines in the

form of serial batch or outsourced with cost. Wang and Cui (2020) considered a robust version of identical parallel machine scheduling with an outsourcing option.

In the job-shop machine environments, Guo and Lei (2014) minimized the sum of the total tardiness and the total outsourcing costs, where a job can be outsourced as a whole to a single subcontractor. Lei and Guo (2016) studied another job-shop scheduling problem to minimize the total tardiness within an outsourcing budget constraint. Safarzadeh and Kianfar (2019) considered job-shop scheduling to minimize the weighted sum of makespan and total outsourcing costs.

There are also some studies of two-machine flowshop scheduling. Qi (2009, 2011) considered three types of outsourcing models to minimize the makespan in the two-machine flowshop: the first model is to outsource both operations of a job to a subcontractor; the second model is to outsource the first job-operations of all jobs to one subcontractor; the third model is to outsource the subset of the first job-operations to one subcontractor. Qi (2009, 2011) considered an outsourcing model where there is only one subcontractor with a single machine for the outsourcing strategy, and the jobs are delivered in a batch to the company from the subcontractor, the outsourcing cost is proportional to the processing time of an outsourced job. In fact the subcontractor is regarded as a single machine that is geographically remote from the company.

Lee and Choi (2011) considered a two-stage production scheduling problem where any operation of two stages in a two-machine flowshop can be outsourced to minimize the weighted sum of the makespan and the total outsourcing costs. The outsourcing cost is proportional to the processing time of the outsourced job. Choi and Chung (2011) and Chung and Choi (2013) analyzed some two-stage flowshop scheduling problems to minimize the sum of the makespan and the total outsourcing costs. These two studies assumed that when a job is outsourced, the both operations of the job should be sent to a subcontractor. Neto and Filho (2011), Tavares Neto and Filho (2011), Mokhtari et al. (2012) and Choi and Park (2014) studied several multi-stage permutation flowshop scheduling problems to minimize the sum of the outsourcing costs and either makespan or total

completion time. These studies also assumed that the entire operations of the job should be sent to a subcontractor.

To the best of our knowledge, this is the first study to consider two-machine flowshop scheduling with outsourcing lead-time. Completing a job including final quality assurance is one of core operations in most manufacturing environment, and hence it is more sensible to consider an outsourcing option on the first operations rather than the second operations. Moreover, this model can be applied to situations where there are multiple subcontractors with different lead-time. The remainder of the paper is organized as following. In Section 2, we provide a detailed problem description as well as the formal notation. Sections 3 and 4 focus on the makespan and total completion time minimization problems for the scheduling costs, respectively. Section 5 contains numerical experiments testing the efficiency of the algorithms, followed by concluding remarks and directions for future research in Section 6.

2. Problem Description

There are n jobs with two serial operations in a two-machine flowshop scheduling environments. The processing time of job j in the first in-house machine is $p_j (j = 1, \ldots, n)$, and if the first operation of job j is outsourced, then the outsourcing cost o_j and the outsourcing lead-time l_j are needed. The outsourcing lead-time includes the production and delivery time of the outside subcontractor, and so the second operation of the outsourced job j cannot start before the outsourcing lead-time l_j. The processing time of the second operation of job j is $q_j (j = 1, \ldots, n)$. Denote by π the set of the outsourced jobs, then the total outsourcing costs can be expressed as $\sum_{j \in \pi} o_j$. The objective of this paper is to minimize the weighted sum of the outsourcing costs and the scheduling measure denoted by either one of makespan and total completion time. While the

scheduling cost is based on time, the outsourcing cost is based on monetary terms. Therefore, the constant δ denotes a weight factor between 0 and 1, and it can be determined by considering the balance of the outsourcing costs and the scheduling measure. The objective function is represented as follow.

Total cost (TC) = $(1-\delta)C_{max} + \delta \sum_{j\in\pi} o_j$ or $(1-\delta)\sum C_j + \delta \sum_{j\in\pi} o_j$

Note that $C_{\max}$ is the makespan, and calculated as $C_{max} = max_{1\le j\le n} C_j$, where C_j is the completion time of job j in the second machine. The constant δ is a weight factor between 0 and 1, which can be determined by considering the balance of the outsourcing costs and the scheduling measure. The standard classification scheme for scheduling problems (Graham et al, 1979) is $\alpha_1|\alpha_2|\alpha_3$ where α_1 describes the machine structure, α_2 gives the job characteristics or restrictive requirements, and α_3 defines the objective function to be minimized. We extend this scheme to provide for the objective function by $(1-\delta)C_{max} + \delta \cdot OC$ or $(1-\delta)\sum C_j + \delta \cdot OC$ where $OC = \sum_{j\in\pi} o_j$. Our problem can be denoted as F2||$(1-\delta)C_{max} + \delta \cdot OC$ and F2||$(1-\delta)\sum C_j + \delta \cdot OC$.

In summary, this study first divides all jobs into outsourced jobs and in-house jobs. By considering that outsourced jobs arrive at the second machine at l_j time, the processing sequence of all the jobs in the two-machine flowshop is determined to minimize the objective cost function. If the first operations of a job is negligible (the processing time is very small or the operation can be missed), then it is optimal to schedule the second operations of the job at time zero (or right after the first negligibly small operation) for the makespan measure. However, for the total completion time measure, it is not trivial to schedule the second operation because the problem is still NP-hard (Garey *et al*, 1976). We now propose several properties of an optimal job schedule that will allow us to develop efficient algorithms. In the following Property 1, the permutation schedule means that the processing order in the first and second machine is identical.

Property 1. The processing order of the in-house jobs follows the permutation schedule in the flowshop system.

Proof. The proof can be easily seen by the pair-wise job interchange arguments. ■

Property 2. In an optimal schedule, there is no intermediate idle time on both the first in-house machine and the second in-house machine.

Proof. There is no reason to have idle time in the first in-house machine for the in-house jobs. Suppose that there exists an intermediate idle time between the operations on the second machine. Then, the intermediate idle time can be removed by delaying all the operations before the idle time without increasing the makespan. ■

3. Minimizing the makespan and the outsourcing costs

This section begins by proving that the problem F2||$(1-\delta)C_{max}+\delta\cdot OC$ is NP-hard, as in Theorem 1.

Theorem 1. The problem F2||$(1-\delta)C_{max}+\delta\cdot OC$ is NP-hard even if all $l_j = 0$.

Proof. The proof can be done in polynomial reduction from the Partition Problem (Garey and Johnson, 1979), which is known to be NP-hard. The Partition Problem is stated as follows;

Given a set $\{a_1,\ldots, a_m\}$ of positive integers, where $\sum_{i=1}^{m} a_i = 2B$ and $0 < a_i < B$, for $i = 1,\ldots, m$, does there exist a subset $S \subset \{1,\ldots, m\}$ such that $\sum_{i\in S} a_i = B$?

Now, consider the following instance of the given problem F2||$(1-\delta)C_{max}+\delta\cdot OC$;

$n = m+1$, $\delta = \frac{1}{B+1}$,

$p_j = a_j$, $q_j = 0$, $o_j = a_j$, $l_j = 0$, for $j = 1,\ldots, m$,

$p_j = 0$, $q_j = B$, $o_j = (B+1)^2$, $l_j = 0$, for $j = m+1$.

Denote by Q a threshold value which is set as

$Q = B$.

Therewith, it will be proved that there exists a feasible schedule π for the problem instance satisfying the relation $(1-\delta)C_{max} + \delta \cdot OC \leq Q$ if and only if there exists a solution to the Partition Problem.

a) For if-part; suppose that there are two disjoint sets X and Y which comprise a solution to the Partition Problem, such as $\{x_1,\ldots,x_{|X|}\}$ and $\{y_1,\ldots,y_{|Y|}\}$, where $x_1+\ldots+x_{|X|} = y_1+\ldots+y_{|Y|} = B$, $\{x_1,\ldots,x_{|X|}, y_1,\ldots,y_{|Y|}\} = \{a_1,\ldots, a_m\}$ and $|X|+|Y|=m$. Then, the associated job sets $X' = \{J_1^X,\ldots,J_{|X|}^X\}$ and $Y' = \{J_1^Y,\ldots,J_{|Y|}^Y\}$ represents the in-house jobs and outsourced jobs, respectively, where $\{J_1^X,\ldots, J_{|X|}^X, J_1^Y,\ldots, J_{|Y|}^Y\} = \{J_1,\ldots, J_m\}$. Now, consider the schedule π as being represented by processing job $m+1$ and all jobs in X' in the in-house machine in arbitrary order, and by outsourcing all jobs in Y'. Thus the associated processing time in X' are $\{x_1,\ldots, x_{|X|}\}$, and the associated outsourcing costs in Y' are $\{y_1,\ldots, y_{|Y|}\}$. Note that $q_j = 0$, $j = 1,\ldots, m$ and $p_{m+1} = 0$, then the completion time in the schedule π is $x_1+\ldots+x_{|X|}=B$ in the first in-house machine and $q_{m+1}=B$ in the second machine. The schedule π is depicted as in Figure 1. The total outsourcing cost in π is $y_1+\ldots+y_{|X|}=B$. The objective cost value of π is $(1-\delta)C_{max} + \delta \cdot OC = (1-\delta)C_{max} + \delta \sum_{j\in\pi} o_j = \left(1 - \frac{1}{B+1}\right) B + \frac{1}{B+1} B = B$ which is equal to Q, satisfying the condition $(1-\delta)C_{max} + \delta \cdot OC \leq Q$.

>> Insert Figure 1 <<

b) For only if-part; suppose that a schedule π is feasible as satisfying the relation $(1-\delta)C_{max} + \delta \cdot OC \leq Q$. Note that the job $m+1$ cannot be outsourced because the outsourcing cost $\delta \cdot o_{m+1} = \frac{1}{B+1}(B+1)^2 = B+1$ is greater than the value Q. Note that $p_{m+1}=0$, $q_{m+1}=B$, and the remaining

jobs satisfy the relations $p_j = a_j$, $q_j = 0$, $o_j = a_j$, $j = 1,\ldots, m$. Denote by $\{J_1^O, J_2^O,\ldots, J_{|O|}^O\}$ the outsourced job set having the outsourcing costs $\{x_1, x_2,\ldots, x_{|O|}\}$ and by $\{J_1^P, J_2^P,\ldots, J_{|P|}^P, J_{m+1}\}$ the in-house processing job set having the processing time $\{y_1, y_2,\ldots, y_{|P|}\}$ in the first machine, respectively, where $\{x_1, x_2,\ldots, x_{|O|}, y_1, y_2,\ldots, y_{|P|}\} = \{a_1, a_2,\ldots, a_m\}$ and $|O|+|P|=m$.

Remind that the relations $\sum_{i=1}^{|O|} x_i + \sum_{i=1}^{|P|} y_i = \sum_{i=1}^{m} a_i = 2B$ hold, and consider the two following cases;

i) the first case $\sum_{i=1}^{|P|} y_i < B$ and $\sum_{i=1}^{|O|} x_i > B$

Suppose that $\sum_{i=1}^{|P|} y_i = B - k$ and $\sum_{i=1}^{|O|} x_i = B + k$, then the makespan of the schedule π is the value B, since the completion time in the second machine is the value $q_{m+1}=B$, and the total outsourcing costs are the value $B + k$. Therefore the total costs of the schedule π is $(1-\delta)C_{max} + \delta \cdot OC = (1-\delta)B + \delta(B+k) = B + \delta \cdot k > B$, which is a contradiction.

ii) the second case $\sum_{i=1}^{|P|} y_i > B$ and $\sum_{i=1}^{|O|} x_i < B$

Suppose that $\sum_{i=1}^{|P|} y_i = B + k$ and $\sum_{i=1}^{|O|} x_i = B - k$. The makespan of the schedule π is the value $B + k$, and the total outsourcing costs are the value $B - k$. Therefore the total costs of the schedule π is $(1-\delta)C_{max} + \delta \cdot OC = \frac{B}{B+1}(B+k) + \frac{1}{B+1}(B-k) = B + \frac{k(B-1)}{B+1} > B$, which is a contradiction.

From the two above cases, we conclude that the relations $\sum_{i=1}^{|O|} x_i = \sum_{i=1}^{|P|} y_i = B$ hold, which means the relation $\sum_{i \in X} a_i = \sum_{i \in Y} a_i = B$ should hold. This implies the existence of a solution to the Partition Problem. ■

The problem F2$||(1-\delta)C_{max} + \delta \cdot OC$ can be expressed as the following mathematical model.

Min $(1-\delta)C_{max} + \delta \sum_{j=1}^{n} o_j(1-y_j)$ (1)

st. $C_{max} \geq C_j$ for all j (2)

$$C_j \geq x_{jk} y_j \left(p_j + \sum_{k'=1}^{k-1} \sum_{j'=1}^{n} p_{j'} x_{j'k'} y_{j'}\right) + q_j \quad \text{for all } j,\ k \quad (3)$$

$$C_j \geq l_j \cdot (1 - y_j) + q_j \quad \text{for all } j \quad (4)$$

$$C_j \geq x_{jk} \sum_{j'=1}^{n} x_{j'k-1} C_{j'} + q_j \quad \text{for all } j,\ k \quad (5)$$

$$\sum_{k=1}^{n} x_{jk} = 1 \quad \text{for all } j \quad (6)$$

$$\sum_{j=1}^{n} x_{jk} = 1 \quad \text{for all } k \quad (7)$$

$$x_{j0} = 0 \quad \text{for all } j$$

$$x_{jk}, y_j \in \{0,1\} \quad \text{for all } j,\ k$$

Let y_j denote the variable having the value 0 if the first operation of job j is outsourced, and 1 otherwise. Let x_{jk} denote the variable having 1 if job j is processed in the kth position in the second machine, and 0 otherwise. Equation (1) represents the objective function of this paper, and Constraints (2) calculate the makespan value of any schedule. Equation (3) means that if job i is produced in-house, the processing in the second machine cannot start before the completion of the processing in the first in-house machine. Equation (4) means that if job j is outsourced, the processing in the second machine cannot start before the outsourcing lead-time. Equation (5) represents that job j can start processing in the second machine after the completion of the previous jobs. Constraints (6) and (7) imply that every job must be processed in the second machine.

We now suggest a way to identify jobs that can be excluded from consideration for outsourcing.

Property 3. If job j satisfies the following condition, job j can be excluded from outsourcing.

$$\delta o_j \geq (1 - \delta)\left(p_j + q_j\right) \quad (7)$$

Proof. If job j is outsourced, the minimum increment of the objective cost is δo_j. Conversely, if job j is processed in-house, the maximum increment of the objective cost is $(1 - \delta)\left(p_j + q_j\right)$. If Eq. (7) holds, it means that the outsourcing cost of job j is too large, so it is more advantageous not

to outsource job j. ■

This paper presents additional properties for the job outsourcing decision. Note that the following Properties 4 and 5 are suggested originally by Lee and Sung (2008a) in a single machine environment, and these properties can be also applied in the two-machine flowshop environment in this paper.

Property 4. (Lee and Sung, 2008a) For any pair of two jobs i and j such that $p_i < p_j$, $o_i \geq o_j$ and $l_i \geq l_j$, it is more cost-beneficial to outsource job j than job i.

Proof. Suppose that there are two jobs i and j such that $p_i < p_j$, $o_i \geq o_j$ and $l_i \geq l_j$. Consider a schedule S where job i is outsourced but job j is not outsourced. Then, the cost of the schedule S is

$$\text{Cost(S)} = \sum_{k\in\pi(S)-\{j\}} o_k + o_j + C_{max}(\text{S}),$$

where $\pi(S)$ is a set of outsourced jobs in the schedule S. Consider another schedule S' where job j is outsourced but job i is not outsourced. Then, the cost of the schedule S' is

$$\text{Cost(S')} = \sum_{k\in\pi(S')-\{i\}} o_k + o_i + C_{max}(\text{S'}).$$

Note that $\pi(S) = \pi(S')$. Since $p_i < p_j$, $o_i \geq o_j$ and $l_i \geq l_j$, it holds that Cost(S) > Cost(S'). Therefore, it is more cost-beneficial to outsource job j than job i. ■

Property 5. (Lee and Sung, 2008a) If $\sum_{i=1}^{n} p_i < min_{1\leq i\leq n}\, l_i$, then it is optimal to process all the jobs on the in-house machine.

Proof. Suppose that job j is outsourced. Then, its outsourcing can make the positive and engative contributions to the objective value. While the associated objective value will be increased due to its outsourcing lead-time and the outsourcing cost, the associated objective value will be decreased due to the associated work saving. The objective value increment is no less than $(1-\delta)\, min_{1\leq i\leq n}\, l_i$, and the decrement is no more than $(1-\delta)p_j$. Since $\sum_{i=1}^{n} p_i < min_{1\leq i\leq n}\, l_i$,

$$(1-\delta)\, min_{1\le i\le n}\, l_i > (1-\delta)\sum_{i=1}^{n} p_i > (1-\delta)p_j,$$

which implies that the increment value is larger than the decrement value. Therefore, it is optimal to process all the jobs on the in-houre machine. ■

The problem in this paper has two decision issues; the first is to classify the in-house jobs and the outsourced jobs, and the second is to determine the processing order of the jobs in the two machine flowshop. If the in-house jobs are already determined, the processing order of the in-house jobs can be determined according to Property 6.

Property 6. The processing order of the in-house jobs can be determined optimally by the Johnson's rule.

Proof. Johnson's rule is a well-known algorithm to find the optimal solution for the problem F2||C_{max}. The detailed proof can be easily seen by the pair-wise job interchange arguments. ■

This section suggests a dynamic programming algorithm based on Property 6. By using the DP algorithm A, we can further investigate the complexity of the considered problem F2||$(1-\delta)C_{max} + \delta \cdot OC$.

DP Algorithm A

---

1 **Input:** $n, p_j, q_j, \delta, o_j, l_j$

2 **Output:** an optimal schedule and its solution value

3 **Code:**

4 *Indexing:* Sequence and index all the jobs in the Johnson's order.

5 *Value Function*: $f_j(x_1, y_1, y_2)$ = the minimum cost of a partial schedule for jobs 1,…, j which are

processed during [0, x_1] in the first in-house machine, and processed during [y_1, y_2] time in the second machine.

6 *Boundary Condition*: $f_0(0, 0, 0) = 0$.

7 *Optimal Solution Value*: $min_{\{(x_1,y_1,y_2)|0\le x_1\le P,0\le y_1\le y_2\le Q\}} f_n(x_1, y_1, y_2)$

where $P = \sum_{j=1}^{n} p_j$, $Q = \sum_{j=1}^{n}(p_j + q_j)$.

8 *Recurrence Function*: $f_j(x_1, y_1, y_2)$ =

$$\begin{cases} \infty, if\ x_1 > \sum_{i=1}^{j} p_i & (i) \\ \infty, if\ y_2 - y_1 < \sum_{i=1}^{j} q_i & (ii) \\ min\begin{cases} min_{\{0\le k\le y_2-q_j-x_1\}}\{f_{j-1}(x_1 - p_j, y_1, y_2 - q_j - k) + (1-\delta)(q_j + k)\}, if\ x_1 \ge p_j\ and\ y_2 - x_1 \ge q_j & (iii) \\ min_{\{0\le k\le y_2-q_j-x_1\}}\{f_{j-1}(x_1, y_1, y_2 - q_j - k)\} + (1-\delta)(q_j + k) + \delta\cdot o_j, if\ y_2 - l_j \ge q_j & (iv) \end{cases} \end{cases}$$

In the recurrence function, the term (*i*) means the situation that there occurs an idle time in the first in-house machine, which violates the optimality condition of Property 2. The term (*ii*) represents an impossible situation, since the time period [y_1, y_2] in the second machine is not enough to process jobs 1,…, *j*. The term (*iii*) represents that job *j* is processed in the first in-house machine, and the term (*iv*) represents that job *j* is outsourced. The maximum number of DP states in DP algorithm A cannot be over nPQ^2, so that the complexity order of DP algorithm A is $O(nPQ^2)$, which is a pseudo-polynomial complexity.

From Property 6, we can see that the processing order of the in-house jobs can be determined as the Johnson's Rule. The outsourced jobs are excluded in the first in-house machine and released at the outsourcing lead-time l_j in the second machine. The processing start-time of outsourced jobs in the second machine can be determined based on Property 7.

Property 7. The in-house jobs have already been sequenced as Johnson's sequence. Let *k* be an outsourced job with lead time l_k and let *RJ* denote a set of jobs starting no sooner than l_k in the pre-

determined Johnson's sequence. Then the processing start-time of job k in the second machine can be determined as follows;

(1) If the second machine is idle at l_k, then job k starts immediately at l_j. If the insertion of job k causes conflicts in other jobs in the second machine, the processing of other jobs are delayed as needed.

(2) If the second machine is processing a preceding job at l_j, then the processing of job k starts immediately after the completion of the preceding job. The processing of jobs in RJ is delayed in the second machine as necessary since job k is interrupted.

Proof. At the time point l_k, the outsourced job k can be considered as an in-house job with $p_k = 0$ and $q_k > 0$, which is the highest priority job in the Johnson's sequence, so job k should be processed first no sooner than l_k. Therefore the processing time of job k in the second machine can be determined as the above (1) and (2). ■

Assume that IS denotes a set of the in-house jobs and OS is a set of the outsourced jobs. If IS and OS are determined, then the remaining schedule can be obtained optimally by the Procedure SEQ based on Property 7, as follows.

Procedure SEQ

---

1 **Input:** IS, OS

2 **Output:** a solution value of the schedule for given IS and OS

3 **Code:**

4 Sequence the jobs in IS as Johnson's sequence and set $a = 1$.

5 Calculate the start-time s_j in the second machine for each job in IS.

6 Select the job with the a-th smallest l_j in OS, which is called as job x for convenience.

7 If all the s_j values are less than l_x, then job x is placed just after the last-positioned job in IS and

added to *IS*, and go to Line 9.

8 Find the job with the smallest s_j value satisfying $s_j \geq l_x$ in *IS*, which is called as job *y*. Job *x* is placed just before job *y* and added to *IS*.

9 If $a = |OS|$, the algorithm is terminated. Otherwise update $a = a + 1$ and go to Line 5.

Now this paper proposes a constructive heuristic that can get solutions efficiently within a very short time, which is the greedy-type. In the heuristic, $\frac{(1-\delta)p_j}{\delta o_j}$ is calculated for each job, and the heuristic idea is that the larger the value, the greater the benefit of outsourcing. The detailed procedure of the heuristic is as follows.

Heuristic HEU

---

1 **Input:** $n, p_j, q_j, \delta, o_j, l_j$

2 **Output:** a schedule and its solution value

3 **Code:**

4 Set *OS* to be empty, and assign all the jobs to *IS*, and calculate the value $\frac{(1-\delta)p_j}{\delta o_j}$ for each job.

5 Calculate the value $\frac{(1-\delta)p_j}{\delta o_j}$ for each job.

6 Find a schedule from Procedure SEQ for the sets *IS* and *OS*, and the objective cost of the obtained schedule is called as S^*, and set $a = 1$.

7 For a = 1 to $|IS|$

8 Select the job with the *a*-th largest $\frac{(1-\delta)p_j}{\delta o_j}$ in *IS*, which is called as job *x*.

9 Find a new schedule by Procedure SEQ for the sets IS_x and OS_x, where $IS_x = IS - \{x\}$ and $OS_x = OS + \{x\}$, and the objective cost of the new schedule is called as S_x.

10 If $S_x < S^*$, then update the sets *IS*, *OS* and S^* with IS_x, OS_x and S_x.

11 Return S^*, *IS* and *OS*.

The optimal solution can be obtained from DP algorithm A, but the DP algorithm takes a lot of time, so this paper proposes a branch-and-bound algorithm to find an optimal solution in a reasonable time.

A node of the B&B algorithm in this paper corresponds to a subset of jobs including the in-house jobs and the outsourced jobs. All the jobs are ordered and indexed by the Johnson's rule, and the B&B algorithm selects the first job in the Johnson's sequence in a node, and adds two new nodes associated with two cases whether to process the job in-house or outsource it. The branching mechanism of the B&B algorithm in this paper is shown in Figure 2, where IS_r and OS_r denote sets of the in-house jobs and the outsourced jobs respectively in node *r*, and UD_r is a set of undetermined jobs in node *r*.

>> Insert Figure 2 <<

The jobs in UD_r are ordered in Johnson's sequence, and the first job *j* in UD_r is selected, and the B&B algorithm creates two new nodes (*r*+1) and (*r*+2) by either one of processing job *j* in-house or outsourcing it. This paper uses the depth-first search for the node selection, which means that the algorithm selects first a node with the smallest number of undetermined jobs in the branching tree. If IS_r and OS_r are given in node *r*, then the production schedule can be determined by Procedure SEQ, and the objective cost of the obtained schedule on node *r* is defined as TC_r. In the algorithm,

the fathoming rules are based on Properties 3, 4 and 5, which means that any node satisfying the conditions from the properties will be fathomed. Denote by *UB* by the solution value from the heuristic HEU, which is used as the initial upper bound. To improve the efficiency of the branch-and-bound search, it is crucial to use a tight lowerbound. The lowerbound value of a node is calculated as follows.

$$LB_r = TC_r + (1-\delta)\sum_{j\in UD_r} q_j \quad (8)$$

The overall procedure of the B&B algorithm is presented as follows.

The branch-and-bound algorithm

---

1 **Input:** $n, p_j, q_j, \delta, o_j, l_j$

2 **Output:** an optimal schedule and UB

3 **Code:**

4 *Initialization*

5 Sequence and index all the jobs as Johnson's sequence.

6 Calculate the value *UB*, and set $r \leftarrow 0,\ IS_r \leftarrow \emptyset,\ OS_r \leftarrow \emptyset,\ UD_r \leftarrow \{1, 2, \ldots., n\}$.

7 *Node selection*

8 In the branching tree, select node *r* from the depth-first search.

9 *Dominance checking*

10 Calculate the lowerbound value LB_r by using equation (8).

11 If $LB_r \geq$ UB then go to Line 20.

12 If any of Properties 3, 4, and 5 holds, then go to Line 20.

13 *Branching*

14 If $UD_r = \emptyset$, then go to Line 18.

15 Select a job *j* with the smallest index in UD_r.

16 Set $IS_{r+1} = IS_r \cup \{j\},\ OS_{r+1} = OS_r,\ UD_{r+1} \leftarrow \{j+1, \ldots., n\}$, and add node (*r*+1) to the

branching tree.

17 Set $IS_{r+2} = IS_r,\ OS_{r+2} = OS_r \cup \{j\},\ UD_{r+2} \leftarrow \{j+1, \ldots., n\}$, and add node ($r$+2) to the branching tree. Go to Line 20.

18 *Update the upperbound*

19 Calculate the objective cost TC_r on node r. If $TC_r < UB$, then set $UB \leftarrow TC_r$.

20 *Node elimination*

21 Delete node r from the branching tree.

22 If the branching tree is empty, then the B&B algorithm is terminated. Otherwise go to Line 7.

---

4. Minimizing the total completion time and the outsourcing costs

Since the problem F2||$\sum C_j$ has been a well-known NP-hard problem, the problem F2$||(1-\delta)\sum C_j + \delta \cdot OC$ is NP-hard. Similarily to the mathematical formulation for F2$||(1-\delta)C_{max} + \delta \cdot OC$, the problem F2$||(1-\delta)\sum C_j + \delta \cdot OC$ can be formulated as follows.

Min $(1-\delta)\sum_{j=1}^{n} C_j + \delta \sum_{j=1}^{n} o_j(1-y_j)$

st.

$C_j \geq x_{jk} y_j \left(p_j + \sum_{k'=1}^{k-1} \sum_{j'=1}^{n} p_{j'} x_{j'k'} y_{j'}\right) + q_j$	for all j, k
$C_j \geq l_j \cdot (1-y_j) + q_j$	for all j
$C_j \geq x_{jk} \sum_{j'=1}^{n} x_{j'k-1} C_{j'} + q_j$	for all j, k
$\sum_{k=1}^{n} x_{jk} = 1$	for all j
$\sum_{j=1}^{n} x_{jk} = 1$	for all k
$x_{j0} = 0$	for all j
$x_{jk}, y_j \in \{0,1\}$	for all j, k

Properties 4 and 5 in Section 2 are still valid for F2||$(1-\delta)\sum C_j + \delta \cdot OC$, and this paper suggests a way to recognize jobs that can be excluded from consideration for outsourcing as follows.

Property 8. If job *j* satisfies the following condition, job *j* can be excluded from outsourcing, where $C_{max}(JS)$ is the makespan value of the Johnson's rule for all the jobs.

$$\delta o_j + (1-\delta)\left(l_j + q_j\right) > (1-\delta)\sum_{i=1}^{n}(p_i + q_i) \quad (9)$$

Proof. If job *j* is outsourced, the minimum increment of the objective cost is $\delta o_j + (1-\delta)\left(l_j + q_j\right)$. Conversely, if job *j* is processed in-house, the completion time of job j cannot exceed the value $(1-\delta)\sum_{i=1}^{n}(p_i + q_i)$. If Eq. (9) holds, it means that the outsourcing cost of job *j* is too large, so it is more advantageous not to outsource job *j*. ■

The problem F2||$(1-\delta)\sum C_j + \delta \cdot OC$ also has two decision issues; the first is to classify the in-house jobs and the outsourced jobs, and the second is to determine the processing order of the jobs in the two machine flowshop. Even if the set *IS* and *OS* have already been determined, it is still not easy to determine optimally the processing order of jobs in *IS*. If *IS* and *OS* are determined, and if the processing order of jobs in *IS* is determined already as any sequence α, then the processing start-time of outsourced jobs in the second machine can be determined from the following Property 9.

Property 9. The in-house jobs have already been sequenced as sequence α. Let *k* be an outsourced job with lead time l_k and let *RJ* denote a set of jobs starting no sooner than l_k in the sequence α. Then the processing start-time of job *k* in the second machine can be determined as follows;

(3) If the second machine is idle at l_k, then job k starts immediately at l_j. If the insertion of job k causes conflicts in other jobs in the second machine, the processing of other jobs are delayed as needed.

(4) If the second machine is processing a preceding job at l_j, then the processing of job k starts immediately after the completion of the preceding job. The processing of jobs in RJ is delayed in the second machine as necessary since job k is interrupted.

Proof. The proof is the same as in Property 7. ■

Assume that all the in-house jobs are sequenced as in a given sequence α. The entire production schedule can be obtained by the following Procedure SEQ(α) based on Property 9.

Procedure SEQ(α)

---

1 **Input:** α, IS, OS

2 **Output:** a schedule and its solution value

3 **Code:**

4 Sequence the jobs in *IS* as the sequence α and set $p = 1$.

5 Calculate the start-time s_j in the second machine for each job in *IS*.

6 Select the job with the a-th smallest l_j in *OS*, which is called as job x for convenience.

7 If all the s_j values are less than l_x, then job x is placed just after the last-positioned job in *IS* and added to *IS*, and go to Line 9.

8 Find the job with the smallest s_j value satisfying $s_j \geq l_x$ in *IS*, which is called as job y. Job x is placed just before job y and added to *IS*.

9 If $a = |OS|$, the algorithm is terminated. Otherwise update $a = a + 1$ and go to Line 5.

Note that the Procedure SEQ(α) depends on the given sequence α. This paper uses the three possible sequencing rules as follows.

1) Johnson's rule

2) SPT: a job with a small $(p_j + q_j)$ is processed firstly.

3) LPT: a job with a large $(p_j + q_j)$ is processed firstly.

Procedure SEQ(α) can be expressed as SEQ(Johnson), SEQ(SPT) and SEQ(LPT) depending on the sequence α. This paper ultimately proposes three heuristics such as HEU(Johnson), HEU(SPT) and HEU(LPT) depending on the sequence α as follows.

HEU(α)

---

1 **Input:** $n, p_j, q_j, \delta, o_j, l_j$

2 **Output:** a schedule and its solution value

3 **Code:**

4 Set *OS* to be empty, and assign all the jobs to *IS*, and calculate the value $\frac{(1-\delta)p_j}{\delta o_j}$ for each job.

5 Find a schedule from Procedure SEQ(α) for the sets *IS* and *OS*, and the objective cost of the obtained schedule is called as S^*, and set $p = 1$.

6 Select a job x with the p-th largest $\frac{(1-\delta)p_j}{\delta o_j}$ value in *IS*. Find a new schedule by Procedure SEQ(α) for the sets IS_x and OS_x, where $IS_x = IS - \{x\}$ and $OS_x = OS + \{x\}$, and the objective cost of the new schedule is called as S_x.

7 If $S_x < S^*$, then update the sets *IS*, *OS* and S^* with IS_x, OS_x and S_x, respectively.

8 If $p = |IS|$, then terminate the algorithm. Otherwise set $p = p+1$ and go to Line 6.

Now this paper proposes a neighborhood exchange heuristic (NEH) in a two-phase approach. The first phase determines the order of processing in-house jobs based on the NEH algorithm of Nawaz et al. (1983). From the review paper by Ruiz and Maroto (2005), the NEH algorithm of Nawaz et al. (1983) has been known to be the best constructive heuristic for the flowshop

scheduling problem. The basic idea is to find a heuristic solution by comparing schedules through inserting a job one-by-one to possible positions. The second phase determines which jobs to be outsourced, which is based on the idea of HEU(α) in this paper. The two-phased approach is called as NEH as follows.

NEH (Neighborhood Exchange Heuristic)

---

1 **Input:** $n, p_j, q_j, \delta, o_j, l_j$

2 **Output:** a schedule and its solution value

3 **Code:**

4 Phase I

5 All the jobs are ordered in ascending order of $(p_j + q_j)$, which is called as sequence Ω.

6 Select the first two jobs in the sequence Ω. Then generate two possible sequences of the two selected jobs and calculate the sum of completion time of the two sequences. Find the best sequence among the two, which is called sequence α, and set $k = 3$.

7 Select k-th job in the sequence Ω, and generate k possible sequences by placing the job in the possible k positions in the sequence α with former jobs, and calculate the sum of completion time of the k sequences. Find the best sequence, and update the sequence α with the new sequence.

8 If $k = n$, then go to PHASE II. Otherwise $k = k + 1$ and go to Line 7.

9 Phase II

10 Set *OS* to be empty, and assign all the jobs to *IS*, and calculate the value $\frac{(1-\delta)p_j}{\delta o_j}$ for each job.

11 Find a schedule from Procedure SEQ(α) for the sets *IS* and *OS*, and the objective cost of the obtained schedule is called as S^*, and set $p = 1$.

12 Select a job x with the p-th largest $\frac{(1-\delta)p_j}{\delta o_j}$ value in *IS*. Find a new schedule by Procedure SEQ(α) for the sets IS_x and OS_x, where $IS_x = IS - \{x\}$ and $OS_x = OS + \{x\}$, and the objective cost of the new schedule is called as S_x.

13 If $S_x < S^*$, then update the sets *IS*, *OS* and S^* with IS_x, OS_x and S_x, respectively.

14 If $p = |IS|$, then terminate the algorithm. Otherwise set $p = p+1$ and go to Line 12

Now this paper proposes a pair-wise exchange heuristic (PEH) in a two-phase approach. The first phase determines the sequence of in-house jobs, and the idea of pair-wise exchange is to find a better schedule by moving forward through exchanging two adjacent jobs in a given sequence. If a better best schedule is found, then the given sequence is updated and go back to the beginning and proceed by moving forward by exchanging two adjacent jobs. The second phase also determines which jobs to be outsourced, which is based on HEU(α). The Procedure PEH (Pair-wise Exchange Heuristic) is presented in detail as follows.

PEH (Pair-wise Exchange Heuristic)

---

1 **Input:** $n, p_j, q_j, \delta, o_j, l_j$

2 **Output:** a schedule and its solution value

3 **Code:**

4 Phase I

5 All the jobs are ordered in ascending order of $(p_j + q_j)$, which is called as sequence α.

6 Set $k = 1$.

7 Select *k*-th and (*k*+1)-th jobs in the sequence α. Exchanging the two jobs in the sequence α, and we can obtain a new schedule α*'*.

8 If α*'* is better than α, then replace α with α*'*, and go to Line 6.

9 If $k = n-1$, then go to PHASE II. Otherwise $k = k + 1$ and go to Line 7.

9 Phase II

10 Set *OS* to be empty, and assign all the jobs to *IS*, and calculate the value $\frac{(1-\delta)p_j}{\delta o_j}$ for each job.

11 Find a schedule from Procedure SEQ(α) for the sets *IS* and *OS*, and the objective cost of the obtained schedule is called as S^*, and set $p = 1$.

12 Select a job x with the p-th largest $\frac{(1-\delta)p_j}{\delta o_j}$ value in IS. Find a new schedule by Procedure SEQ(α) for the sets IS_x and OS_x, where $IS_x = IS - \{x\}$ and $OS_x = OS + \{x\}$, and the objective cost of the new schedule is called as S_x.

13 If $S_x < S^*$, then update the sets IS, OS and S^* with IS_x, OS_x and S_x, respectively.

14 If $p = |IS|$, then terminate the algorithm. Otherwise set $p = p+1$ and go to Line 12

This paper derives a B&B algorithm for the problem. The depth-first search is used for the node selection in the B&B algorithm. The B&B algorithm creates two new nodes for each undetermined job in a node by either one of processing in-house or outsourcing the job. For example, if there are $|UD_r|$ nodes in node r, then $2|UD_r|$ nodes are newly created and added to the B&B tree, which is depicted as in Figure 3.

>> Insert Figure 3 <<

In the algorithm, the fathoming rules are based on Properties 4, 5 and 8. Denote by UB by the solution value from the Procedure NEH, which is used as the initial upper bound. To improve the efficiency of the branch-and-bound search, the lowerbound value of a node is calculated as follows, where $q_{[j]}$ is the j-th smallest processing time in the set UD_r.

$$LB_r = TC_r + (1-\delta)\sum_{j=1}^{|UD_r|}(|UD_r| - j + 1)\, q_{[j]} \qquad (9)$$

The overall procedure of the B&B algorithm is presented as follows.

The Branch-and-Bound algorithm

---

1 **Input:** $n, p_j, q_j, \delta, o_j, l_j$

2 **Output:** an optimal schedule and UB

3 **Code:**

4 *Initialization*

5 Calculate the value *UB* from the Procedure NEH.

6 Sequence and index all the jobs as in the upperbound.

7 Set $r \leftarrow 0,\ IS_r \leftarrow \emptyset,\ OS_r \leftarrow \emptyset,\ UD_r \leftarrow \{1, 2, \ldots., n\}$.

8 *Node selection*

9 In the branching tree, select node *r* from the depth-first search.

10 *Dominance checking*

11 Calculate the lowerbound value LB_r by using equation (9).

12 If $LB_r \geq$ UB then go to Line 22.

13 If any of Properties 4, and 5 holds, then go to Line 22.

14 *Branching*

15 If $UD_r = \emptyset$, then go to Line 20. Otherwise set $p = 1$.

16 Select the p-th job in UD_r, which is called as job j.

17 Set $IS_{r+(2p-1)} = IS_r \cup \{j\},\ OS_{r+(2p-1)} = OS_r,\ UD_{r+(2p-1)} \leftarrow \{j+1, \ldots., n\}$, and add node ($r$+2$p$-1) to the branching tree.

18 Set $IS_{r+2p} = IS_r,\ OS_{r+2p} = OS_r \cup \{j\},\ UD_{r+2p} \leftarrow \{j+1, \ldots., n\}$, and add node ($r$+2$p$) to the branching tree.

19 If $p = |UD_r|$, then go to Line 22. Otherwise set $p = p+1$, and go to Line 16.

20 *Update the upperbound*

21 Calculate the objective cost TC_r on node r. If $TC_r < UB$, then set $UB \leftarrow TC_r$.

22 *Node elimination*

23 Delete node *r* from the branching tree.

24 If the branching tree is empty, then the B&B algorithm is terminated. Otherwise go to Line 8.

---

5. Computational Experiments

In this section, we conduct extensive numerical tests to analyze the performance of the DP, heuristics, and branch-and-bound algorithms proposed in Section 2 and 3. The algorithms are coded in C++, and run on a personal computer with Intel (R) Core (TM) i5-4590 processor with CPU 3.3GHz clock and 16GB RAM.

Since there is no computational study which considers the setting addressed in this paper in the existing literature, we have adapted the data generation scheme used in Lee and Sung (2008a, 2008b). In Lee and Sung (2008a, 2008b), the authors study a single machine problem with an option of outsourcing. We evaluate the performance of the proposed algorithms by considering the impact of: number of jobs (n), processing time (p_j and q_j), cost parameter (δ), outsourcing cost (o_j), and outsourcing lead-time (l_j). The parameters are set as follows.

1) The processing time p_j and q_j are generated from U[1, 30], where U[a, b] is an uniform distribution between a and b.
2) The cost parameter δ is generated from U[0.3, 0.7].
3) The outsourcing cost o_j is generated from U[1, 10].
4) The outsourcing lead-time l_j is generated from $\mathrm{U}[0.1P,\ 0.3P]$, where $\mathrm{P} = \sum_{j=1}^{n} p_j$.

Now for the first problem $\mathrm{F2}||(1-\delta)C_{max} + \delta \cdot OC$, the number of jobs is set to 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70 75, 80, and 20 problem instances are randomly generated for each job. For DP algorithm A in Table 1, "Aver. states" denotes the average number of states and "Aver. time (sec)" denotes the average time until the optimal solution is obtained, and NO denotes the number of problems that the associated algorithm finds the optimal solution within 3600 seconds. For the B&B algorithm, "Aver. nodes" denotes the number of nodes until the optimal solution is obtained, and "Aver. Out (%)" denotes the percentage of the outsourced jobs in the schedule obtained by the associated algorithm. For the heuristic HEU, "Aver. Gap (%)" denotes the average

gap (%) which is calculated by the following equation, where *OPT* is the optimal solution from the B&B algorithm and Sol_{Heu} is a solution value obtained through the heuristic HEU.

$$\text{Gap}(\%) = \frac{(Sol_{Heu} - OPT)}{OPT} \times 100.$$

The implications we can find from Table 1 are as follows. DP algorithm A was able to find the optimal solutions for up to 25 jobs instances within 3,600 seconds. Since the DP algorithm does not utilize the dominance properties or fathoming rules, it can be said that the number of 25 jobs is not a small number. The B&B algorithm can solve up to 80 jobs problems within 3600 seconds, which is an exceptional performance. For an NP-hard problem in the scheduling research area, it is not easy for the B&B algorithm to solve the problems with large job-size. The reason why the performance of the DP algorithm and B&B algorithm is so outstanding in this paper is probably because the algorithms start with the Johnson's rule, which seems to reduce drastically unnecessary searches. On average, less than 2% of jobs were outsourced in the optimal solutions by the B&B algorithm, which means that the outsourcing cost is relatively large compared to the processing time. It can be seen that the heuristic HEU has an average GAP (%) of 3% or less, which seems to be very good performance as a constructive heuristic of the Greedy-type. Moreover, the GAP (%) of the heuristic solutions decreased as the number of jobs increased, which is because the outsourcing percentage approaches the optimal outsourcing percentage as the number of jobs increases.

>> Insert Table 1 <<

The performance of the algorithms was analyzed for the fluctuation of the outsourcing cost. o_j is generated from U[10, 30], U[20, 40] and U[30, 50], and the computational results are summarized in Table 2. Even though the outsourcing cost changes, the performance of the heuristic HEU and the B&B algorithm does not seem to change significantly, and the outsourcing percentage also

doesn't seem to fluctuate significantly. The authors speculated that if the outsourcing cost increased, the outsourcing percentage would decrease, but actual experiments did not confirm such results.

>> Insert Table 2 <<

The performance of the algorithms was analyzed for the fluctuation of the delta δ, which is generated from U[0.3, 0,4], U[0,4, 0.5], U[0,5, 0.6] and U[0,6, 0.7], and the computational results are summarized in Table 3. Even though the delta value changes, the performance of the heuristic HEU and the B&B algorithm does not seem to change significantly. When the delta δ is at U[0.4, 0.5], the outsourcing percentage increases and appears to decrease in the rest.

>> Insert Table 3 <<

The performance of the algorithms was analyzed for the fluctuation of the processing time p_j in the first in-house machine, which are generated from U[10, 20], U[20, 30] and U[30, 40], and the computational results are summarized in Table 4. The result shows that as the processing time in the first in-house machine increases, the performance of the B&B algorithm and the heuristic decreases. This is because if the processing time in the first in-house machine is zero or very small, then our problem becomes a single machine scheduling problem to minimize Makespan or the total completion time which can be solved in polynomial time, and this implies that the complexity of the problem may increase as the processing time in the first in-house machine increases. It can be observed that the outsourcing percentage increased significantly as the first processing time increases.

>> Insert Table 4 <<

For the second problem F2$||(1-\delta)\sum C_j + \delta \cdot OC$, we now analyze the performance of the proposed heuristics and the B&B algorithm. For the test, the number of jobs is set to 5, 10, 15, 20, 25, 30, 35, and 20 problem instances are randomly generated for each case. The results of experiments on the performance of the derived algorithms are summarized as in Table 5. The result shows that B&B algorithm can find an optimal solution up to 35 jobs within 3600 seconds. Compared to other Scheduling researches, finding the optimal solutions up to 35 jobs can be said a considerable number. We can guess that the upperbound, lowerbound, and branching mechanisms of this work are effective. In the heuristic performance, HEU(SPT) performed best on 5-job instances, but NEH performed best on the rest. This can be thought of as effective job sequencing in Phase I in NEH.

>> Insert Table 5 <<

The performance of the algorithms was analyzed for the fluctuation of the outsourcing cost. o_j is generated from U[10, 30], U[20, 40] and U[30, 50], and the computational results are summarized in Table 6. The performance evaluation was conducted only on HEU(SPT), NEH, and PEH, which appear to be performing well. It can be said that the greater the o_j value, the greater the problem complexity. This is because the time for the B&B algorithm to find the optimal solutions increases. The more o_j increases, the more complex the problem becomes, but the average Gap (%) in heuristics tends to decrease. Among them, NEH shows the best performance. As o_j grows, the outsourcing ratio of all algorithms decreases, which can be said to be a natural result. Regardless of the value of OC, NEH is showing the best performance.

>> Insert Table 6 <<

The performance of the algorithms was analyzed for the fluctuation of the delta δ, which is generated from U[0.3, 0,4], U[0,4, 0.5], U[0,5, 0.6] and U[0,6, 0.7], and the computational results are summarized in Table 7. The overall experimental results for the value δ show similarities to those for OC. This is because an increase in the value of δ equals an increase in the outsourcing cost. Table 7 shows that the increasing value of δ increases the time for the B&B algorithm to find the optimal solutions, but decreases the solution gap of heuristics. It can also be observed that the outsourcing ratio decreases as δ increases, and among the three heuristics, NEH has the best performance.

>> Insert Table 7 <<

The performance of the algorithms was analyzed for the fluctuation of the processing time p_j in the first in-house machine, which are generated from U[10, 20], U[20, 30] and U[30, 40], and the computational results are summarized in Table 8. Processing time p_j and outsourcing cost o_j have conflicting relationships. This is because the advantage of outsourcing increases when the processing time p_j increases relatively, which means the outsourcing rate tends to increase. It can be said that the larger the p_j value, the higher the problem complexity, because the B&B algorithm takes more time to find the optimal solutions. Although NEH was the best of the heuristics, the performance of all heuristics tends to be similar as p_j increases. This can be interpreted as a result of the decrease in the importance of in-house job sequencing in Phase I as the out-sourcing jobs increase.

>> Insert Table 8 <<

6. Conclusion

This paper considers two stage flowshop scheduling to choose the in-house processing or outsourcing in the first machine. It is assumed that there may be multiple subcontractors, and the outsourced jobs are delivered at the outsourcing lead-time from subcontractors, and processed and completed in the second in-house machine. This paper considers two types of scheduling problems; the first problem is to minimize the weighted sum of the outsourcing costs and the makespan and the second problem is to minimize the weighted sum of the outsourcing costs and the total completion time.

For the first problem, this paper analyzes the problem complexity, and derives some solution properties, and proposes a DP algorithm, a constructive heuristic and a B&B algorithm. The DP algorithm was able to find the optimal solutions up to the problems of 25 jobs and the B&B algorithm can solve very large size problems of 80 jobs within 3600 seconds. In the two-machine flowshop, since the job sequence is fixed by the Johnson's rule, unnecessary searches are reduced significantly, and so the performance of the B&B algorithm is considered to be excellent. It can also be seen that the heuristic HEU has an average GAP (%) of 3% or less which can be said to be very good performance as a constructive heuristic of Greedy-type.

For the second problem, this paper derives some heuristic algorithms such HEU(Johnson), HEU(SPT), HEU(LPT), NEH and PEH, and a B&B algorithm. The HEU(Johnson), HEU(SPT) and HEU(LPT) are constructive heuristics of the Greedy-type, but NEH and PEH are heuristics based on the ideas of neighborhood search and pair-wise exchange, respectively. The B&B algorithm can solve very large size problems of 35 jobs within 3600 seconds, and Among the five heuristics, we can see that NEH shows the best performance.

Future research may focus on studying outsourcing lead-time in various multiple machine environments with different scheduling criteria, such as total weighted completion time and total

tardiness. Also, it would be interesting to develop other heuristics based on meta-heuristic approaches such as Genetic Algorithm and compare performace of the heuristics in various settings.

References

Choi, B.-C., and M.-J. Park. 2014. “Outsourcing Decisions in m-Machine Permutation Flow Shop Scheduling Problems with Machine-Dependent Processing Times.” Asia-Pacific Journal of Operational Research 31 (4): 1450028.

Choi, B. C., and J. Chung. 2011. “Two-machine flow shop scheduling problem with an outsourcing option.” European Journal of Operational Research 213(1): 66-72.

Chung, D. Y., and B. C. Choi. 2013. “Outsourcing and scheduling for two-machine ordered flow shop scheduling problems.” European Journal of Operational Research 226: 46–52.

Drucker, Peter F. 1989. "Sell the Mailroom", Wall Street Journal.

Garey, M.R., D.S. Johnson and R. Sethi. 1976. “The complexity of flowshop and jobshop scheduling.” Mathematics of Operations Research 1(2): 117-129.

Guo, X., and D. Lei. 2014. “Bi-Objective Job Shop Scheduling with Outsourcing Options.” International Journal of Production Research 52: 3832–3841.

Hong, J. M., and J. H. Lee. 2016. “Outsourcing Decisions in Single Machine Scheduling Problem with Multiple External Facilities.” Journal of Marine Science and Technology 24: 603–609.

Lee K., and B. C. Choi. 2011. “Two-stage production scheduling with an outsourcing option.” European Journal of Operational Research 213: 489–497.

Lee, I. S., and C. S. Sung. 2008a. "Single Machine Scheduling with Outsourcing Allowed." International Journal of Production Economics 111: 623-634.

Lee, I. S., and C. S. Sung. 2008b. "Minimizing Due Date Related Measures for a Single Machine Scheduling with Outsourcing Allowed." European Journal of Operational Research 186: 931-952.

Lei, D., and X. Guo. 2016. “A Shuffled Frog-Leaping Algorithm for Job Shop Scheduling with Outsourcing Options.” International Journal of Production Research 54: 4793–4804.

Liao, B., Q. Song, J. Pei, S. Yang and P. M. Pardalos. 2020. "Parallel-machine group scheduling with inclusive processing set restrictions, outsourcing option and serial-batching under the effect

of step-deterioration." Journal of Global Optimization 78: 717–742.

Liu, Z., W.-C. Lee, and J.-Y. Wang. 2016. “Resource Consumption Minimization with a Constraint of Maximum Tardiness on Parallel Machines.” Computers & Industrial Engineering 97: 191–201.

Lu, L., L. Zhang, J. Zhang and L. Zuo. 2020. "Single Machine Scheduling with Outsourcing Under Different Fill Rates or Quantity Discount Rates." Asia-Pacific Journal of Operational Research 37: 1950033.

Mokhtari, H., and I. N. K. Abadi. 2013. “Scheduling with an Outsourcing Option on Both Manufacturer and Subcontractors.” Computers & Operations Research 40: 1234–1242.

Mokhtari, H., I. N. K. Abadi, and M. R. Amin-Naseri. 2012. “Production Scheduling with Outsourcing Scenarios: A Mixed Integer Programming and Efficient Solution Procedure.” International Journal of Production Research 50: 5372–5395.

Nawaz, M., E. E. Enscore Jr., and I. Ham. 1983. “A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem.” Omega 11(1): 91–95.

Neto, R. F. T., M. G. Filho, and F. M. da Silva. 2015. “An Ant Colony Optimization Approach for the Parallel Machine Scheduling Problem with Outsourcing Allowed.” Journal of Intelligent Manufacturing 26: 527–538.

Qi, X. 2009. “Two-stage production scheduling with an option of outsourcing from a remote supplier.” Journal of System Science and System Engineering 18: 1-15.

Qi, X. 2011. “Outsourcing and production scheduling for a two-stage flowshop.” International Journal of Production Economics 129; 43–50.

Ruiz, R., and C. Maroto. 2005. “A comprehensive review and evaluation of permutation flowshop heuristics.” European Journal of Operational Research 165: 479–494.

Safarzadeh, H., and F. Kianfar. 2019. "Job shop scheduling with the option of jobs outsourcing." International Journal of Production Research 54: 3255–3272.

Suliman, S. 2000. “A two-phase heuristic approach to the permutation flowshop scheduling

problem.” International Journal of Production Economics 64: 143–152.

Tavares Neto, R.F., and M. G. Filho. 2011. “An ant colony optimization approach to a permutational flowshop scheduling problem with outsourcing allowed.” Computers & Operations Research 38: 1286–1293.

Wang, S. and W. Cui. 2020. "Approximation algorithms for the min-max regret identical parallel machine scheduling problem with outsourcing and uncertain processing time." International Journal of Production Research, DOI: 10.1080/00207543.2020.1766721.

Zhong, W., and Z. Huo. 2013. “Single Machine Scheduling Problems with Subcontracting Options.” Journal of Combinatorial Optimization 26: 489–498.

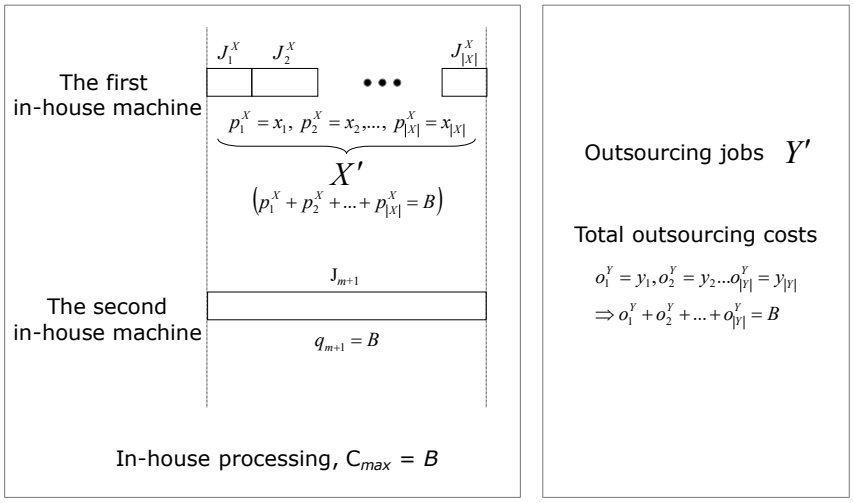

Figure 1. The graphic representation of the schedule π.

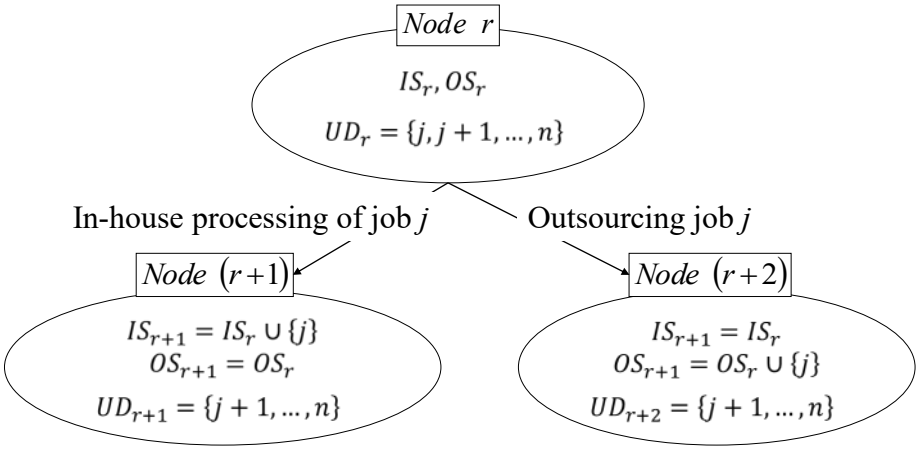

Figure 2. The branching mechanism of the B&B algorithm.

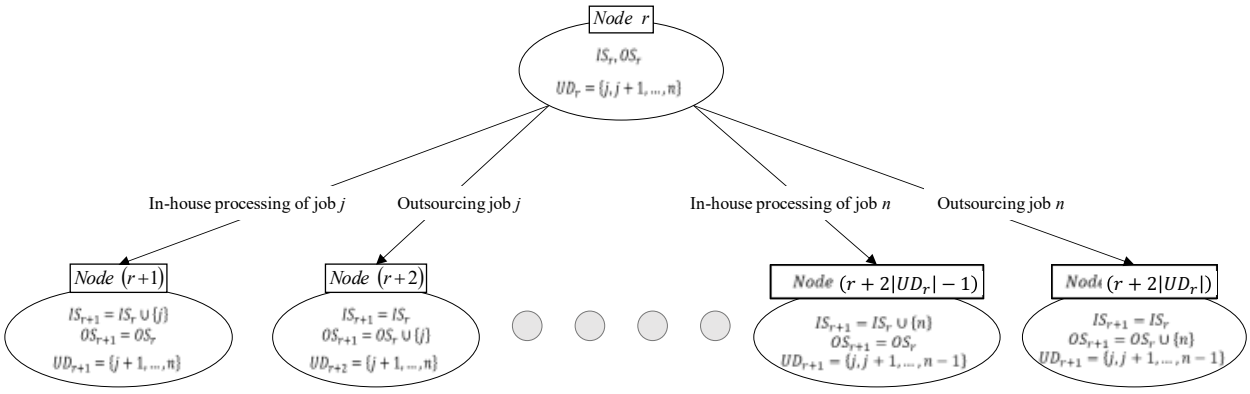

Figure 3. The branching mechanism of the B&B algorithm.

< Table 1. The computational experiments on the proposed algorithms >

Job	DP algorithm A			B&B algorithm				HEU	
	Aver. states	Aver. Time (sec)	No	Aver. nodes	Aver. Time (sec)	Aver. Out. (%)	No	Aver. Gap (%)	Aver. Out. (%)
5	62	0.0	20	17	0.0	2.00	20	2.94	12.00
10	2046	0.0	20	55	0.0	1.00	20	0.57	8.50
15	65534	0.1	20	169	0.0	2.00	20	1.86	8.67
20	2097150	3.9	20	350	0.0	0.75	20	0.58	7.50
25	67108862	165.5	20	345	0.0	0.60	20	0.26	6.00
30	(*)	(*)	(*)	113570	0.2	0.00	20	0.38	6.17
35	(*)	(*)	(*)	2440	0.0	0.00	20	0.00	8.43
40	(*)	(*)	(*)	105013	0.2	1.63	20	0.27	6.00
45	(*)	(*)	(*)	5858	0.0	1.11	20	0.19	4.44
50	(*)	(*)	(*)	69178	0.1	1.80	20	0.25	9.40
55	(*)	(*)	(*)	12464	0.0	0.00	20	0.00	2.73
60	(*)	(*)	(*)	519132	1.7	0.42	20	0.10	7.33
65	(*)	(*)	(*)	109381	0.3	0.00	20	0.00	3.00
70	(*)	(*)	(*)	1677784	6.5	0.71	20	0.22	3.64
75	(*)	(*)	(*)	475794	1.8	0.53	20	0.12	4.20
80	(*)	(*)	(*)	66988860	323.5	1.62	19	0.06	4.19

(*) means the problem instances which DP algorithm A cannot find the optimal solutions within 3600 seconds.

< Table 2. The computational experiments of the algorithms according to the outsourcing costs >

o_j	Job	B&B algorithm			HEU	
		Aver. Nodes	Aver. Time (sec)	Aver Out. (%)	Aver. Gap (%)	Aver Out. (%)
[10, 30]	10	93	0.00	7.00	0.10	8.00
	30	641	0.01	4.25	0.02	4.25
	50	20807	0.06	5.67	0.02	5.67
	Aver.	7180	0.02	5.64	0.05	5.97
[20, 40]	10	19	0.01	1.50	0.03	1.50
	30	21421	0.05	6.75	0.06	7.00
	50	30147	0.09	4.00	0.01	4.17
	Aver.	17195	0.05	4.08	0.03	4.22
[30, 50]	10	15	0.00	1.50	0.00	1.50
	30	7427	0.02	7.25	0.06	7.25
	50	7482	0.02	2.17	0.00	2.17
	Aver.	4975	0.01	3.64	0.02	3.64
[100, 150]	10	10	0.00	0.00	0.00	0.00
	30	31	0.00	0.00	0.00	0.00
	50	54	0.00	0.00	0.00	0.00
	Aver.	31	0.00	0.00	0.00	0.00
[500, 750]	10	10	0.00	0.00	0.00	0.00
	30	30	0.00	0.00	0.00	0.00
	50	50	0.00	0.00	0.00	0.00
	Aver.	30	0.00	0.00	0.00	0.00

< Table 3. The computational experiments of the algorithms according to the δ value >

δ	Job	B&B algorithm			HEU	
		Aver. nodes	Aver. Time (sec)	Aver Out. (%)	Aver. Gap (%)	Aver Out. (%)
[0.01, 0.1]	10	1304	0.02	12.50	0.11	11.00
	30	114249220	385.26	4.75	0.00	5.25
	50	196625898	969.04	6.83	0.00	6.83
	Aver.	103625474	451.44	8.03	0.04	7.69
[0.3, 0.4]	10	184	0.01	12.00	0.11	13.50
	30	450	0.01	2.75	0.01	3.00
	50	2968	0.02	4.67	0.00	4.67
	Aver.	1201	0.01	6.47	0.04	7.06
[0.4, 0.5]	10	122	0.01	14.50	0.19	14.50
	30	2749	0.01	5.50	0.04	5.50
	50	22887	0.07	6.33	0.00	6.33
	Aver.	8586	0.03	8.78	0.08	8.78
[0.5, 0.6]	10	60	0.00	8.00	0.17	9.00
	30	1325	0.00	6.50	0.07	6.50
	50	4176	0.01	5.50	0.01	5.67
	Aver.	1853	0.00	6.67	0.08	7.06
[0.6, 0.7]	10	31	0.00	6.50	0.15	7.00
	30	4661	0.01	6.00	0.02	6.25
	50	16306	0.04	5.83	0.01	6.00
	Aver.	6999	0.02	6.11	0.06	6.42
[0.9, 0.99]	10	11	0.00	2.00	0.00	2.00
	30	238	0.00	4.00	0.00	4.00
	50	112	0.00	1.00	0.00	1.00
	Aver.	120	0.00	2.33	0.00	2.33

< Table 4. The computational experiments of the algorithms according to the processing times >

p_j	Job	B&B algorithm			HEU	
		Aver. nodes	Aver. Time (sec)	Aver Out. (%)	Aver. Gap (%)	Aver Out. (%)
[10, 20]	10	313	0.00	15.00	0.37	17.00
	30	16428	0.05	17.25	0.11	17.50
	50	760663	1.95	16.00	0.09	16.17
	Aver.	259135	0.67	16.08	0.19	16.89
[20, 30]	10	538	0.01	45.00	1.85	46.50
	30	84514	0.18	48.50	1.40	51.00
	50	36802298	103.10	47.17	1.01	49.00
	Aver.	12295783	34.43	46.89	1.42	48.83
[30, 40]	10	448	0.00	59.50	4.10	60.00
	30	70907	0.15	59.75	3.46	64.00
	50	19604611	54.10	59.00	1.85	62.50
	Aver.	6558655	18.08	59.42	3.14	62.17

< Table 5. The computational experiments on the proposed algorithms >

Job	B&B algorithm				HEU(Johnson)		HEU(SPT)		HEU(LPT)		NEH		PEH	
	Aver. nodes	Aver. Time (sec)	Aver. Out. (%)	No	Aver. Gap (%)	Aver. Out. (%)	Aver. Gap (%)	Aver. Out. (%)	Aver. Gap (%)	Aver. Out. (%)	Aver. Gap (%)	Aver. Out. (%)	Aver. Gap (%)	Aver. Out. (%)
5	22	0.01	29.0	20	11.0	53.0	2.9	34.0	18.9	68.0	3.5	32.0	3.2	30.0
10	261	0.00	22.5	20	11.3	48.5	5.1	29.5	31.7	62.0	4.2	28.5	4.9	28.0
15	1830	0.02	22.6	20	13.9	51.0	6.8	34.6	36.6	63.0	5.2	27.6	6.5	34.6
20	14579	0.18	21.2	20	16.3	55.0	8.3	31.0	37.6	63.5	5.9	30.7	7.2	31.5
25	106443	2.63	18.6	20	14.1	51.6	7.0	30.2	34.2	63.4	4.4	25.8	7.0	30.4
30	2532345	108.16	20.5	20	15.9	55.1	8.0	34.6	34.8	63.8	5.8	28.1	8.2	34.5
35	10889726	765.29	18.4	20	15.4	55.8	7.4	30.8	31.0	65.8	4.7	25.1	6.4	31.1

< Table 6. The computational experiments of the algorithms according to the outsourcing costs >

o_j	Job	B&B algorithm			HEU(SPT)		NEH		PEH	
		Aver. nodes	Aver. Time (sec)	Aver Out. (%)	Aver. Gap (%)	Aver. Out. (%)	Aver. Gap (%)	Aver. Out. (%)	Aver. Gap (%)	Aver. Out. (%)
[10, 30]	10	750	0.0	19.5	6.9	25.5	4.8	20.0	6.4	23.5
	20	240457	3.3	16.3	8.3	23.5	6.0	15.0	8.5	21.5
	30	40440739	1544.4	18.8	8.4	27.0	4.9	22.5	7.9	28.0
	Aver.	13560648	515.9	18.2	7.9	25.3	5.2	19.2	7.6	24.3
[20, 40]	10	1417	0.0	13.5	7.1	19.0	4.7	12.5	5.7	18.0
	20	9110402	102.1	19.0	9.0	26.0	7.0	21.0	8.4	25.3
	30	38500173	1492.7	13.0	6.9	20.7	3.6	14.7	6.5	21.2
	Aver.	15870664	531.6	15.2	7.7	21.9	5.1	16.1	6.9	21.5
[30, 50]	10	648	0.0	8.5	7.7	14.5	3.9	10.0	6.1	13.5
	20	4006392	44.5	15.3	8.2	19.5	5.7	16.2	8.1	19.5
	30	62629197	2321.4	13.5	6.6	20.0	3.4	16.3	6.3	19.8
	Aver.	22212079	788.6	12.4	7.5	18.0	4.3	14.2	6.8	17.6

< Table 7. The computational experiments of the algorithms according to the δ value >

δ	Job	B&B algorithm			HEU(SPT)		NEH		PEH	
		Aver. nodes	Aver. Time (sec)	Aver Out. (%)	Aver. Gap (%)	Aver. Out. (%)	Aver. Gap (%)	Aver. Out. (%)	Aver. Gap (%)	Aver. Out. (%)
[0.3, 0.4]	10	259	0.0	27.0	6.7	41.5	6.1	39.0	8.4	42.5
	20	7654	0.1	21.8	8.3	33.2	5.8	30.5	7.5	34.5
	30	444574	21.0	22.5	8.0	38.2	5.5	35.3	7.9	35.2
	Aver.	150829	7.0	23.8	7.7	37.6	5.8	34.9	7.9	37.4
[0.4, 0.5]	10	241	0.0	25.5	6.2	34.5	4.0	28.5	5.7	33.0
	20	6558	0.1	18.8	7.5	34.0	4.6	24.0	7.1	32.5
	30	2654507	117.5	19.0	7.8	33.2	4.5	25.3	7.2	31.8
	Aver.	887102	39.2	21.1	7.2	33.9	4.4	25.9	6.7	32.4
[0.5, 0.6]	10	266	0.0	24.0	6.2	32.0	5.5	28.0	6.9	31.0
	20	11424	0.2	22.8	7.1	32.5	5.8	29.5	7.9	31.0
	30	903182	40.8	18.0	8.1	30.0	5.5	26.2	7.8	30.5
	Aver.	304957	13.7	21.6	7.1	31.5	5.6	27.9	7.5	30.8
[0.6, 0.7]	10	326	0.0	20.5	6.4	31.0	5.5	24.5	5.6	27.5
	20	17942	0.3	21.3	8.1	32.5	5.5	27.0	6.9	30.8
	30	10205147	438.4	17.0	8.2	31.0	4.8	24.7	8.2	30.0
	Aver.	3407805	146.2	19.6	7.6	31.5	5.3	25.4	6.9	29.4

< Table 8. The computational experiments of the algorithms according to the processing times >

p_j	Job	B&B algorithm			HEU(SPT)		NEH		PEH	
		Aver. nodes	Aver. Time (sec)	Aver Out. (%)	Aver. Gap (%)	Aver. Out. (%)	Aver. Gap (%)	Aver. Out. (%)	Aver. Gap (%)	Aver. Out. (%)
[10, 20]	10	193	0.0	28.5	4.6	38.0	5.4	40.5	4.9	37.5
	20	12995	0.2	24.0	10.6	31.0	8.2	32.3	9.8	32.5
	30	903218	40.2	20.0	9.5	32.8	5.8	31.0	9.1	33.3
	Aver.	305468	13.5	24.2	8.2	33.9	6.5	34.6	7.9	34.4
[20, 30]	10	266	0.0	38.5	5.6	52.0	7.1	54.5	5.8	52.5
	20	54349	0.6	41.8	12.0	54.0	11.4	60.3	11.7	54.2
	30	19254796	793.2	38.2	15.1	49.3	13.5	59.7	15.2	50.7
	Aver.	6436470	264.6	39.5	10.9	51.8	10.7	58.2	10.9	52.5
[30, 40]	10	329	0.0	57.0	7.4	62.0	8.6	69.0	7.1	62.5
	20	347630	4.1	54.8	12.5	63.0	12.6	71.3	12.9	61.0
	30	56103903	2007.8	52.3	14.3	59.0	12.5	67.3	14.5	58.3
	Aver.	18817287	670.6	54.7	11.4	61.3	11.2	69.2	11.5	60.6