Journal of Marine Science and Technology

Volume 24 | Issue 3 Article 25

OUTSOURCING DECISIONS IN SINGLE MACHINE SCHEDULING PROBLEM WITH MULTIPLE EXTERNAL FACILITIES

Jung Man Hong
LG-CNS, Seoul, Republic of Korea, icjhlee@inje.ac.kr

Jong Hyup Lee
Department of Electronic, Telecommunications, Mechanical & Automotive Engineering, Inje University, Gimhae, Republic of Korea.

Recommended Citation

Hong, Jung Man and Lee, Jong Hyup (2016) "OUTSOURCING DECISIONS IN SINGLE MACHINE SCHEDULING PROBLEM WITH MULTIPLE EXTERNAL FACILITIES," *Journal of Marine Science and Technology*: Vol. 24 : Iss. 3 , Article 25.
DOI: 10.6119/JMST-015-1216-1
Available at: https://jmstt.ntou.edu.tw/journal/vol24/iss3/25

Journal of Marine Science and Technology, Vol. 24, No. 3, pp. 603-609 (2016)
DOI: 10.6119/JMST-015-1216-1

OUTSOURCING DECISIONS IN SINGLE MACHINE SCHEDULING PROBLEM WITH MULTIPLE EXTERNAL FACILITIES

Jung Man Hong[1] and Jong Hyup Lee[2]

Key words: outsourcing, single machine scheduling, due date, pseudo-polynomial.

ABSTRACT

This paper considers outsourcing decisions in a scheduling problem. The objective of the problem is to minimize the outsourcing costs under the constraints of the capacity of outsourcing facilities and due date of each job. The problem is composed of three kinds of decisions. The first decision is the selection of jobs to be processed in-house, the second is to schedule the in-house processing jobs with due date constraints, and the last is to select an outsourcing provider for each outsourced job where the outsourcing provider has a capacity constraint. Some optimality conditions and solution properties for the problem are presented. A solution algorithm with pseudo-polynomial complexity is suggested to find the optimal solution of the problem. The main contributions of this paper are as follows: (1) the mathematical model for the problem is proposed, (2) the pseudo-polynomial algorithm, Depth First Search (DFS) is developed to find the optimal solution and (3) several optimality properties for the problem are addressed. Numerical experiments show that the DFS algorithm has better results than Dynamic Programming (DP) in efficiency even for larger numbers of jobs and outsourcing providers.

I. INTRODUCTION

The proposed problem is to find the optimal schedule with outsourcing in which jobs are processed in external facilities to improve the overall scheduling quality. The objective of the problem is to minimize the total outsourcing cost under the constraints of the capacity of external facilities and the due date of each job.

The scheduling problem with outsourcing would be considered in a project in which a large volume of resource have to be allocated to complete the project. One such project is a ship building in which it takes several years to complete it. The due date of such project is very important because the violation of the due date would result in a large amount of cost or penalty. In the planning of this kind of project, one of the most difficult considerations is various resource requirements over the entire period of the project. (for example, man power, work place, and facilities, etc.) It is not efficient to have all the resource for the project to resolve the fluctuation of the resource requirements. It can be better if some of the resources can be outsourced to optimize business operations. One of the other cases for such project is a software development in which a large volume of manpower is required. In this project, there are precedence considerations among the groups of jobs. Each group of jobs has its own due date to satisfy overall due date requirements. To meet the due date and promote cost competitiveness, outside resources with any economic price can be used. Therefore, the outsourcing strategy is one of the options that decision makers can consider to improve competitiveness in business.

Outsourcing has attracted a lot of interest in recent years. McCarthy and Anagnostou (2004) studied the rationale of outsourcing from an economic viewpoints. Cachon and Harker (2002) and Kaipia and Tanskanen (2003) investigated why many companies consider outsourcing as an attractive strategy to alleviate price competitiveness. Kim (2003) examined the problems of selecting outsourcing resources (companies). Kolisch (2000) addressed a make-to-order production problem with outsourcing, for which he has derived a mixed-integer programming model to minimize holding and setup costs of the associated supply chain. Lee et al. (2002) presented a model for an advanced planning and scheduling (APS) problem with outsourcing, for which they have derived a genetic algorithm to minimize the makespan. Lee and Sung (2008) suggested outsourcing scheduling problems with various objective functions. Various scheduling problems with outsourcing in different multi-stage production systems have been considered (Choi and Chung, 2011; Lee and Choi, 2011; Neto and Filho, 2011; Qi, 2011; Mokhtari et al., 2012). Mishra et al. (2008) developed a Tabu-simulated annealing method for a mixed-integer pro-

Paper submitted 09/24/14; revised 11/18/15; accepted 12/16/15. *Author for correspondence: Jong Hyup Lee (e-mail: icjhlee@inje.ac.kr).*
[1] *LG-CNS, Seoul, Republic of Korea.*
[2] *Department of Electronic, Telecommunications, Mechanical & Automotive Engineering, Inje University, Gimhae, Republic of Korea.*

gramming model of integrated planning and scheduling with outsourcing allowed. Coman and Ronen (2000) formulated the problem as a Linear Programming problem and identified an analytical solution. Chung et al. (2005) and Chen and Li (2008) considered the scheduling problem with subcontracting options. They have suggested a heuristic to solve the problem.

Most of the previous researches assumed that there was only one outsourcing provider. As a result, the decision problem was to choose between insourcing and outsourcing manufacturing. But in real world business, more than two outsourcing providers would be considered because each of outsourcing providers has its own advantages and disadvantages. Therefore, the scheduling problem with outsourcing should be considered with multiple choice of outsourcing providers so that the overall cost saving can be achieved taking advantage of such competition among outsourcing providers. Although many studies have considered the outsourcing scheduling problems, there are few papers that have taken multiple options of outsourcing into consideration (Feng et al., 2011; Li and Wan, 2014).

In this paper, we consider the single machine scheduling problem with multiple external facilities (outsourcing). The problem is composed of three kinds of decisions. The first decision is the selection of jobs to be processed in-house, the second is to schedule the in-house processing jobs with due date constraints, and the last is to select an outsourcing provider for each outsourced job where the outsourcing provider has a capacity constraint. The objective function of the problem is to minimize the total outsourcing cost.

The proposed problem can be described as in the following: We are given a set of jobs and a set of outsourcing providers, J and K, respectively. A processing time and a due date of job j in set J, p_j and d_j, are given. An outsourcing cost of job j to k in set K, o_{jk}, is also given. Let π denote a schedule and O_π^k be a set of outsourced jobs to provider k in schedule π. Then the total outsourcing cost is $\sum_{k\in K}\sum_{j\in O_\pi^k} o_{jk}$. The outsourcing provider has a capacity constraint such that each outsourced job j requires r_{jk} resource for outsourcing provider k and the sum of the resource used by provider k is limited by the capacity R_k. We assume that the due date constraints are always satisfied for outsourced jobs, such as $C_\pi^j \le d_j$ for $\forall j \in O_\pi$, where C_π^j denotes the completion time of job j with a schedule π. Without loss of generality, it is assumed that p_j, d_j, and o_{jk} are integer values and all the jobs are available from time zero, that is, the arrival time of each job assumed to be zero. Then, the problem can be formulated as follows:

$$\text{minimize } \sum_{k\in K}\sum_{j\in O_\pi^k} o_{jk}$$

subject to

$$C_\pi^j - d_j \le 0, \forall j \in J \text{ and } \sum_{j\in O_\pi^k} \le R_k, \forall k \in K.$$

In the next section, the problem complexity and several optimal properties are developed.

II. THE PROBLEM COMPLEXITY AND SOLUTION PROPERTIES

Scheduling problems can be represented as $\alpha|\beta|\gamma$ by using the standard classification scheme of scheduling research. α denotes the shop (machine) environment, β denotes the shop condition, and γ contains the objective function to be optimized. The proposed scheduling problem can be categorized as a single machine problem to minimize the outsourcing cost with the maximum lateness constraints. Accordingly, the proposed problem is expressed as $1\left|C_j \le d_j\right|\sum_{k\in K}\sum_{j\in O_\pi^k} o_{jk}$. The problem $1\left|C_j \le d_j\right|\sum_{k\in K}\sum_{j\in O_\pi^k} o_{jk}$ is Non-deterministic Polynomial time (NP)-hard, so it cannot be solved efficiently (Hong and Lee, 2013).

Before we begin to develop a solution algorithm, several optimality properties are presented.

Proposition 1. For the problem $1\left|C_j \le d_j\right|\sum_{k\in K}\sum_{j\in O_\pi^k} O_{jk}$, there exists an optimal schedule in which all the non-outsourced jobs are sequenced in EDD (earliest due date) order.

Proof.:

It can be proven with an interchange argument. Suppose that there exists an optimal schedule $\pi^* = (\ldots, j_l, \ldots, j_k, \ldots)$ such that jobs in set $J\backslash O_{\pi^*}$ are not sequenced in EDD order. For simplicity of notation, let us assume that l-th and k-th jobs in set $J\backslash O_{\pi^*}$ satisfy $l < k$ and $d_l \ge d_k$. And from the optimality assumption of schedule π^*, the inequalities $C_{\pi^*}^l \le C_{\pi^*}^k$, $C_{\pi^*}^l \le d_l$ and $C_{\pi^*}^k \le d_k$ are satisfied. If two jobs, j_l and j_k, are interchanged, the resulting schedule $\pi_1^* = (\ldots, j_k, \ldots, j_l, \ldots)$ is also feasible because of the inequality $C_{\pi_1^*}^k \le C_{\pi_1^*}^l = C_{\pi^*}^k \le d_k \le d_l$. From this inequality, the feasibility conditions of the schedule π_1^*, $C_{\pi_1^*}^k \le d_k$ and $C_{\pi_1^*}^k \le d_l$, are satisfied. The schedule π_1^* is also optimal because the objective function value of π_1^* is the same as that of the schedule π^*.

This completes the proof.

From Proposition 1, we assume that all the jobs are indexed in EDD order without loss of generality. Thus, if $d_i \le d_j$, then $i < j$.

For Proposition 2, let NO_π be the set of non-outsourced jobs for a schedule π. Then the completion time for non-outsourced job j with a schedule π, C_π^j, can be expressed as $\sum_{i<j,\, i\in NO_\pi}(p_i + \omega_i) + p_j$ where $\omega_i \ge 0$ be the idle time after i-th non-outsourced job in any schedule π.

Proposition 2. For the problem $1\left|C_j \le d_j\right|\sum_{k\in K}\sum_{j\in O_\pi^k} o_{jk}$, there exists an optimal schedule in which all the non-outsourced (i.e., in-house) jobs are sequenced with no idle time.

Proof.:

In the optimal schedule π^*, the constrains $\sum_{i<j,\, i\in NO_{\pi^*}}(p_i+\omega_i)+p_j \le d_j$ will be satisfied. Then the schedule with $\omega_i = 0$ also satisfies the constraints and has the same objective function value.

This complete the proof.

Corollary 1. From Proposition 1 and 2, the problem can be reduced to both the selection of jobs to be outsourced and the assignment of the outsourced jobs to an outsourcing provider.

Proof.:

Corollary 1 can be easily proven, so the proof is omitted.

Let decision variable y_{jk} represents if the job j would be assigned to the outsourcing provider k. The variable y_{jk} will have 1 if the job j is assigned to the outsourcing provider k. Otherwise, y_{jk} will be 0. Then, the integer programming model for the problem is as follows:

$$\text{minimize } \sum_{k\in K}\sum_{j\in J} o_{jk} y_{jk},$$

subject to

$$\sum_{i<j} p_i(1-\sum_{k\in K} y_{ik}) + p_j \le d_j + M\sum_{k\in K} y_{jk},\ \forall j\in J$$

$$\sum_{j\in J} r_{jk}\ y_{jk} \le R_k,\quad \forall k\in K$$

$$\sum_{k\in K} y_{jk} \le 1,\quad \forall j\in J$$

$$y_{jk}\in\{0,1\},\quad \forall j\in J, k\in K,$$

where M represents any big value.

III. SOLUTION PROCEDURE FOR THE PROBLEM

In this section, a network modeling of the problem is suggested and a search procedure in the network model is also considered to find the optimal solution with efficiency.

To make a network model of the problem, several definitions are needed. Let $\pi(i)$ be a partial schedule for jobs 1, 2, …, i, where $i \in J$. Using the definition of $\pi(i)$, the problem $1\left|C_j \le d_j\right|\sum_{k\in K}\sum_{j\in O_\pi^k} o_{jk}$ can be modeled as a network problem and solved by applying a search algorithm. For the network model of the problem, all the possible combinations of (i, m, r_1, …, $r_{|K|}$) denotes the nodes in the network where m, r_1, …, $r_{|K|}$ are any integers satisfying $0 \le m \le \sum_{j\in J} p_j$, $r_1 \le R_1$, …, $r_{|K|} \le R_{|K|}$. The network modeling of the problem starts with the root node, (0, 0, 0, …, 0). Links and nodes are added into the network at each iteration of the network modeling. From the root node, links and nodes are generated to make the whole network model of the problem. If we have a node (i, m, r_1, …, $r_{|K|}$) in the network model, then there can be 1 + |K| links from this node such as (i + 1, m + p_{i+1}, r_1, …, $r_{|K|}$), (i + 1, m, r_1 + $r_{(i+1)(1)}$, …, $r_{|K|}$), …, (i + 1, m, r_1, …, $r_{|K|}$ + $r_{(i+1)(|K|)}$), respectively. If the inequality $(m + p_{i+1}) \le d_{i+1}$ is satisfied, then there is a link from (i, m, r_1, …, $r_{|K|}$) to (i + 1, m + p_{i+1}, r_1, …, $r_{|K|}$) and the weight of the link is zero. If the inequality $r_k + r_{(i+1)(|k|)} \le R_k$ is satisfied for $k \in K$, then there is a link from (i, m, r_1, …, $r_{|K|}$) to (i + 1, m, r_1, …, $r_{|K|}$ + $r_{(i+1)(|K|)}$) and the weight of the link is $o_{(i+1)(k)}$. The maximum number of links from each node to others is to be |K| + 1. Note that r_k in the node (i, m, r_1, …, r_k, …, $r_{|K|}$) denotes the capacity that has been consumed at outsourcing provider k at node (i, m, r_1, …, r_k, …, $r_{|K|}$).

From the following procedure, we can generate a network model for the problem $1\left|C_j \le d_j\right|\sum_{k\in K}\sum_{j\in O_\pi^k} o_{jk}$.

Network generation procedure

1. Let a set of nodes N_1 and a set of links L be empty, i.e. $N_1 = \varnothing$ and $L = \varnothing$. And let a set of nodes N_2 have a root node, initially, i.e. $N_2 = \{(0, 0, 0, …, 0)\}$.
2. Select an arbitrary node in the set N_2 and let (i, m, r_1, …, $r_{|K|}$) be the selected node. If $N_2 = \{(0, 0, 0, …, 0)\}$, then (0, 0, 0, …, 0) will be the selected node.
3. If the inequality $(m + p_{i+1}) \le d_{i+1}$ is satisfied for the selected node, then go to step 4. Otherwise go to step 5.
4. The link from (i, m, r_1, …, $r_{|K|}$) to (i + 1, m + p_{i+1}, r_1, …, $r_{|K|}$) is added into the set of links L with its weight of zero. The node (i + 1, m + p_{i+1}, r_1, …, $r_{|K|}$) is added to the set N_2.
5. Iterate steps 6 and 7 for all $k \in K$.
6. If the inequality $(r_k + r_{(i+1)(k)}) \le R_k$ is satisfied for $k \in K$, then go to step 7. Otherwise, perform step 6 again with another $k \in K$.
7. The link from (i, m, r_1, …, $r_{|K|}$) to (i + 1, m, r_1, …, r_k + $r_{(i+1)(k)}$, …, $r_{|K|}$) is added into L and its weight is $o_{(i+1)(k)}$. The node (i + 1, m, r_1, …, r_k + $r_{(i+1)(k)}$, …, $r_{|K|}$) is added into N_2.
8. If all the iterations of steps 6 and 7 for the selected node (i, m, r_1, …, $r_{|K|}$) are completed, then the node (i, m, r_1, …, $r_{|K|}$) is added into N_1 and subtracted from N_2.
9. If the set N_2 is not empty yet, go to step 2.

Fig. 1 is the flow diagram of the network generation procedure. Let $G\left(1\left|C_j \le d_j\right|\sum_{k\in K}\sum_{j\in O_\pi^k} o_{jk}\right)$ be the network generated with problem $1\left|C_j \le d_j\right|\sum_{k\in K}\sum_{j\in O_\pi^k} o_{jk}$ by applying the procedure depicted in Fig. 1. Note that the network $G\left(1\left|C_j \le d_j\right|\sum_{k\in K}\sum_{j\in O_\pi^k} o_{jk}\right)$ takes the shape of a tree. In G(·), a path from the root node, (0, 0, 0, …, 0), to any end node constructs a feasible schedule and the sum of the link weights on the path is the sum of the outsourcing cost of the corresponding schedule. A solution algorithm that searches all the paths from the root node to the end nodes in the tree G(·) can be considered. The algorithm calculates all the shortest

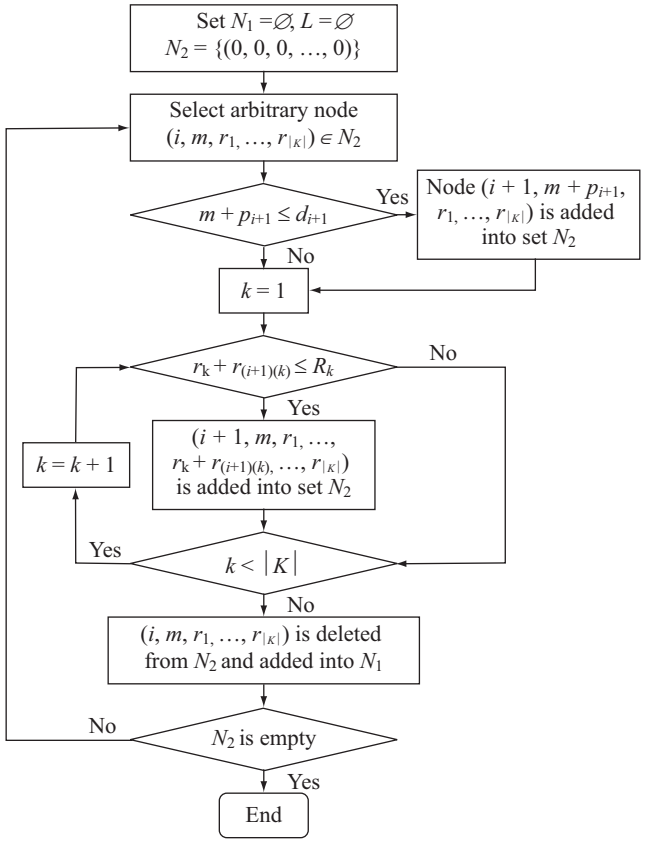

Fig. 1. Diagram of the network generation procedure.

paths to all of end nodes in the network and the minimum value among the shortest paths to all the end nodes is the optimal value of the problem. Due to the fact that the total number of paths in the tree G(·) cannot exceed the value $(|K|+1)^{|J|}$, the optimal solution for problem $1|C_j \le d_j|\sum_{k\in K}\sum_{j\in O_\pi^k} o_{jk}$ can be found in the exponential time complexity with the order of $(|K|+1)^{|J|}$ by applying a search algorithm. However, because of the exponential complexity of the search algorithm, some modifications are needed to improve the efficiency.

In the network modeling of the problem, all the end nodes of the network (tree) should be explored to find the optimal solution. To make this search procedure to be more effective, Depth First Search (DFS) procedure is suggested for the graph $G\left(1|C_j \le d_j|\sum_{k\in K}\sum_{j\in O_\pi^k} o_{jk}\right)$. DFS starts with the root node, (0, 0, 0, …, 0). A node will be selected as a next search node among the nodes linked to (0, 0, 0, …, 0). At each iteration of DFS, the link from $(i, m, r_1, \ldots, r_{|K|})$ to $(i+1, m+p_{i+1}, r_1, \ldots, r_{|K|})$ has a higher priority than any other links when selecting a node for the next search. Let a solution *S*(*DFS*) be the first feasible schedule found by DFS and the set $O_{S(DFS)}$ be the subset of jobs that are selected to be outsourced under the schedule *S*(*DFS*). The set $NO_{S(DFS)}$ is the set of jobs that are selected to be processed in the in-house machine on the schedule *S*(*DFS*). We assume that all the jobs in the set $J = O_{S(DFS)} \cup NO_{S(DFS)}$ are indexed in EDD order. $|J|$ is the number of jobs in the set *J*. Then the pseudo-code using the recursive function dfs(·) for the DFS is as follows:

Procedure DFS

set *job.number* = 0, *best.solution* = MAX,
call function *dfs*(*job.number*),

function *dfs*(*job.number*)

if (*job.number* > |*J*|), then return
if (*'the total outsourcing cost so far'* > *best.solution*), then return
update *'the total outsourcing cost so far'* with π(*job.number*)
update *best.solution*,
if ($job.mp + p_{job.number+1} \le d_{job.number+1}$),
  then
    update π(*job.number* + 1)
    call function *dfs*(*job.number* + 1)
for (*k* = 1; *k* <= |*K*|; *k*++)
  if $\left(r^k_{\pi(job.number)} + r_{(job.number+1)(k)} \le R_k\right)$,
    then
      update π(*job.number* + 1)
      call function *dfs*(*job.number* + 1)

In Procedure DFS, *job.number* denotes the current depth of the search and MAX represents any big value. The procedure DFS starts with *dfs*(1), where *job.number* equals 1. '*the total outsourcing cost so far*' denotes the sum of the outsourcing cost from job 1 to *job.number*. *best.solution* indicates the best solution found so far. $r^k_{\pi(job.number)}$ denotes $\sum_{j\in O_\pi^k} r_{jk}$. Lastly, *job.mp* represents C^j_π. Let $\pi(\cdot)$ be the first feasible schedule found by applying Procedure DFS. Then some optimality conditions for the problem are suggested as in the following propositions.

Proposition 3. If $|O_{\pi(\cdot)}| \le 1$ for the problem with a constant outsourcing cost ($o_{jk} = o$) for $\forall j \in J$ and $\forall k \in K$ and with arbitrary due date d_j for $\forall j \in J$, then the schedule $\pi(\cdot)$ is the optimal solution.

Proof.:

(1) If $|O_{\pi(\cdot)}| = 0$, then the schedule $\pi(\cdot)$ is the optimal.

(2) In case of $|O_{\pi(\cdot)}| = 1$, let the job *j* be the element of the set $O_{\pi(\cdot)}$, which means the condition $\sum_{i\le j} p_i > d_j$ is satisfied. Then we have two cases at the optimal solution. The first one is at least one more jobs not in the outsourced group should be added to the outsourced group. The second is that job *j* should be interchanged with the precedent job *i*, $d_i < d_j$ to make a feasible schedule which has no outsourced job. In the first case in which at least one more

job should be outsourced, the schedule $\pi(\cdot)$ is optimal because $o_{jk} = o$ for $\forall j \in J$ and $\forall k \in K$. In the second case, if we assume that there is the feasible schedule π' of $|O_{\pi'}| = 0$, then (from Proposition 1) there should exist an optimal schedule such that all the jobs are sequenced in EDD order, which contradicts that the schedule π' is feasible.

This completes the proof.

Proposition 4. For the problem with constant processing time ($p_j = p$) for $\forall j \in J$ and with arbitrary due date d_j for $\forall j \in J$, the schedule $\pi(\cdot)$ always gives the minimum number of outsourced jobs among all the feasible solutions.

Proof.:

For the problem with constant processing time ($p_j = p$) for $\forall j \in J$, the schedule $\pi(\cdot)$ in which all the jobs satisfying the condition $\sum_{i<j,\, i\in NO_{\pi(\cdot)}} p + p \le d_j$ are in the set of non-outsourced jobs gives the schedule with the minimum number of outsourced jobs.

This completes the proof.

With arbitrary outsourcing cost, o_{jk}, the schedule with the minimum number of outsourced jobs does not always give the optimal schedule. In Proposition 5, the optimality condition for the schedule with the minimum number of outsourced jobs is presented.

Proposition 5. For the problem with constant processing time and outsourcing cost ($p_j = p$, $o_{jk} = o$) for $\forall j \in J$ and $\forall k \in K$ and with arbitrary due date d_j for $\forall j \in J$, the schedule $\pi(\cdot)$ always gives the optimal solution.

Proof.:

From Proposition 4, the schedule $\pi(\cdot)$ gives the minimum number of outsourced jobs among all the feasible solutions. Because ($o_{jk} = o$) for $\forall j \in J$ and $\forall k \in K$, this gives the optimality conditions.

This completes the proof.

To compare the efficiency of Procedure DFS, a DP (Dynamic Programming) algorithm, which is known to be more efficient than a general search algorithm, is derived for the problem. Let a value function $f(i, m, r_1, \ldots, r_{|K|})$ be the minimum length of the path from the root node to a node $(i, m, r_1, \ldots, r_{|K|})$ and a set $V(G(\cdot))$ be the set of nodes in network $G(\cdot)$. Then the following recurrence relation can be derived. The value function $f(i, m, r_1, \ldots, r_{|K|})$ is a minimum value among $f(i\text{-}1, m\text{-}p_j, r_1, \ldots, r_{|K|})$ and $f(i\text{-}1, m, r_1, \ldots, r_k\text{-}r_{jk}, \ldots, r_{|K|}) + o_{jk}$ for all $k \in K$ and $(i\text{-}1, m, r_1, \ldots, r_k\text{-}r_{jk}, \ldots, r_{|K|}) \in V(G(\cdot))$. The optimal value is $\min_{\left\{0\le m\le \sum_{j\in J} p_j,\, 0\le r_t\le R_t,\, \forall t\in K\right\}} \left\{ f\left|J\right|, m, r_1, \ldots, r_k, \ldots, r_{|K|} \right\}$ where $(|J|, m, r_1, \ldots, r_k, \ldots, r_{|K|}) \in V(G(\cdot))$. The DP algorithm for the problem $1\left|C_j \le d_j\right|\sum_{k\in K}\sum_{j\in O_\pi^k} o_{jk}$ can be summarized as follows:

Dynamic Programming DP

1. Set $f(|J|, m, R_1, \ldots, R_{|K|}) = 0$ for all non-negative integers m less than or equal to $\min\left\{d_{|J|}, \sum_{j\in J} p_j\right\}$.
2. For the i-th iteration, the value function $f(i, m, r_1, \ldots, r_{|K|})$ can be calculated as follows:

$$f(i, m, r_1, \ldots, r_{|K|}) =$$

$$\min\begin{cases} f(i-1, m-p_j, r_1, \ldots, r_{|K|}), \text{ if } m-p_j \ge 0 \text{ and } m \le d_j, \\ f(i-1, m, r_1, \ldots, r_k - r_{jk}, \ldots, r_{|K|}) + o_{jk}, \text{ if } r_k - r_{jk} \ge 0, \\ \infty \end{cases}$$

Because the total number of states of the Procedure DFS cannot exceed the value $\left|J\right| \times \sum_{j\in J} p_j \times R_1 \times \ldots \times R_{|K|}$, an optimal solution for the problem can be found in the pseudo-polynomial complexity order of $O\left(\left|J\right| \times \sum_{j\in J} p_j \times R_1 \times \ldots \times R_{|K|}\right)$. The complexity of the DP algorithm is the same as that of Procedure DFS. Two solution procedures, Procedure DFS and DP algorithm, have the same complexity because these two procedures search the optimal solution based on the same network model. However, the efficiency of them would differ because they search the optimal solution with different search methods. The efficiency of these two solution procedures are tested and compared to each other in the next section.

IV. NUMERICAL EXPERIMENTS

In this section, the performances of the suggested procedure are evaluated. Computational results of the algorithm are presented with randomly generated problems. The parameters of the problems are generated with the predefined statistical distributions. The statistical distributions of the test instances are selected in the similar way as in the references (Koulamas, 1994; Choi et al., 2005; Lee and Sung, 2008). The processing time p_j and the outsourcing cost o_{kj} are generated from the uniform distribution between 1 and 10, and between 1 and 30, respectively. The due date d_j is randomly generated from a uniform distribution depending on the sum of the processing time and two parameters Span of Due Date (SDD) and Tardiness Factor (TF). SDD denotes the degree of span of due dates, which means that the larger value of SDD will result in the wider range of d_j's value. TF denotes the tardiness factor which is approximately proportional to the ratio of jobs likely to be tardy (Raman and Talbot, 1993). We assume that d_j has a uniform distribution between $P_{sum} \times$ (1-TF-SDD/2) and $P_{sum} \times$ (1-TF+SDD/2), where $P_{sum} = \sum_{j\in J} p_j$. The difficulties of the problems may be determined by the values of SDD and TF (Lee and Sung, 2008). In our experiments, SDD and TF are selected among 0.2, 0.4, 0.6, 0.8, and 1.0. For each combination of SDD and TF, 30 instances are generated. The capacity

Table 1. Computational results for DP and DFS algorithm with $|J| = 10$ and $|K| = 2$.

SDD	TF	Computational time of DP	Computational time of DFS	Minimum objective function value
0.2	0.2	0.0001	0.0000	2.5556
0.2	0.4	0.0002	0.0000	4.0000
0.2	0.6	0.0001	0.0000	2.1333
0.2	0.8	0.0001	0.0000	1.1176
0.2	1.0	0.0001	0.0000	0.9444
0.4	0.2	0.0007	0.0001	6.7368
0.4	0.4	0.0007	0.0000	4.7391
0.4	0.6	0.0008	0.0000	7.8571
0.4	0.8	0.0006	0.0000	7.3448
0.4	1.0	0.0006	0.0000	6.4643
0.6	0.2	0.0023	0.0000	12.6667
0.6	0.4	0.0024	0.0001	11.3125
0.6	0.6	0.0026	0.0000	16.9474
0.6	0.8	0.0023	0.0000	16.9286
0.6	1.0	0.0017	0.0000	13.4286
0.8	0.2	0.0055	0.0000	27.6154
0.8	0.4	0.0044	0.0000	22.7556
0.8	0.6	0.0050	0.0000	21.0909
0.8	0.8	0.0042	0.0000	20.5306
0.8	1.0	0.0036	0.0000	20.5918
1.0	0.2	0.0070	0.0000	31.4286
1.0	0.4	0.0071	0.0000	26.2917
1.0	0.6	0.0073	0.0000	23.3061
1.0	0.8	0.0072	0.0000	23.8800
1.0	1.0	0.0067	0.0000	21.3200

Table 2. Computational results for DP and DFS algorithm with $|J| = 15$ and $|K| = 2$.

SDD	TF	Computational time of DP	Computational time of DFS	Minimum objective function value
0.2	0.2	0.0015	0.0000	2.5000
0.2	0.4	0.0026	0.0000	0.7273
0.2	0.6	0.0016	0.0000	1.1176
0.2	0.8	0.0016	0.0000	3.4091
0.2	1.0	0.0018	0.0000	2.5789
0.4	0.2	0.0736	0.0000	10.7000
0.4	0.4	0.0989	0.0000	5.6818
0.4	0.6	0.0812	0.0000	7.1200
0.4	0.8	0.0607	0.0000	7.4667
0.4	1.0	0.0707	0.0000	9.6857
0.6	0.2	0.4699	0.0002	14.0000
0.6	0.4	0.4113	0.0001	21.8056
0.6	0.6	0.4033	0.0000	19.5122
0.6	0.8	0.3502	0.0000	19.3333
0.6	1.0	0.4800	0.0000	24.6136
0.8	0.2	1.2399	0.0003	26.3333
0.8	0.4	2.0197	0.0001	29.2500
0.8	0.6	1.3816	0.0000	26.0667
0.8	0.8	1.2379	0.0001	34.5102
0.8	1.0	1.2843	0.0000	32.7917
1.0	0.2	2.6899	0.0006	47.2500
1.0	0.4	3.5021	0.0001	38.4286
1.0	0.6	3.4959	0.0001	33.2600
1.0	0.8	3.3675	0.0000	31.2000
1.0	1.0	3.8294	0.0000	33.4000

of k-th external facility, i.e., R_k, is set to $CF \times \sum_{j \in J} r_{jk}$, where CF is the capacity factor that is selected among 0.2, 0.4, 0.6, 0.8, and 1.0.

For the numerical experiments, *C* language was used for implementing the suggested algorithms. The machine with Core i-5 CPU with 4G memory was used for running the numerical experiments. In Table 1, the experimental results for Procedure DFS (or DFS algorithm) and Dynamic Programming DP (or DP) are presented. Computational times are measured in seconds.

From Table 1, we can show that the DFS algorithm has better results than DP in efficiency; the DFS algorithm can solve larger sized problems. In Table 1, it is noted that the efficiency of the suggested algorithms are significantly affected by the number of jobs and the degree of SDD, but relatively less affected by the TF.

The computational results in Table 2 show that the number of jobs in the problem considerably affects the efficiency of DP, but not really affects the suggested algorithm, DFS. From Tables 1 and 2, it is noted that the outsourcing cost (i.e. the objective function value of the problem) is more affected by SDD than TF.

For each $|J|$ and $|K|$ values, various SDD and TF values are tested as in Tables 1 and 2. Table 3 represents the computational results of DFS algorithm for larger numbers of jobs and outsourcing providers. The computational times in column b and the minimum objective function values in column c in Table 3 are the average values of the results executed with various SDD and TF values.

In Table 3, it is observed that almost all of the computational times of DFS algorithm are reasonably short even for large numbers of jobs and outsourcing providers. The efficiency of the suggested algorithm are affected by the both the number of jobs and the number of outsourcing providers.

As shown in Table 3, it is reasonable to assume that the cost will be decreased as the number of alternatives in the problem increases. For example, the minimum objective function value with $|J| = 30$ and $|K| = 2$ is 34.4448 in contrast to 27.1571 in case of $|J| = 30$ and $|K| = 3$. It is noted that the outsourcing cost (i.e. the objective function value of the problem) decreases as the number of outsourcing providers is increased. The same can be applied to the real world problem. If we have more choice for the outsourcing providers, the cost will be decreased. That is the motivation for us to consider the scheduling problem with multiple outsourcing providers.

Table 3. Computational results for DFS algorithm with various $|J|$ and $|K|$ values.

$\|J\|$	$\|K\|$	Computational time of DFS	Minimum objective function value
10	2	0.0000	14.1595
15	2	0.0001	19.3097
20	2	0.0041	26.6097
30	2	0.2582	34.4448
20	3	0.0033	19.2985
30	3	1.6955	27.1571
40	3	108.8124	35.5917

V. CONCLUSIONS

In this paper, we have studied the single machine scheduling problem with multiple external facilities. The problem is to minimize the total outsourcing cost subject to the outsourcing capacity and the due date constraints. Several optimality properties have been presented. To solve the problem, we have developed the DSF algorithm efficiently searching the network converted from the problem.

The computational experiments of the proposed scheme have been performed for randomly generated problems. The proposed procedure has illustrated outstanding performance in comparison to DP, even for larger numbers of jobs and outsourcing providers.

The outsourcing is one of the business strategies to improve the competitiveness of the company. As the competition increases in the global economy, the importance of the outsourcing strategy will grow. The scheduling problem with outsourcing and also with additional real world constraints can be considered for further research subjects. For example, the logistics cost constraints for the outsourcing or the precedence constraints among jobs can be addressed as a next step. Non-zero arrival times of jobs can also be considered in the future work.

REFERENCES

Cachon, G. P. and P. T. Harker (2002). Competition and outsourcing with scale economies. Management Science 48(10), 1314-1333.

Chen, Z.-L. and C.-L. Li (2008). Scheduling with subcontracting options. IIE Transactions 40(12), 1171-1184.

Choi, S.-W, Y.-D. Kim and G.-C. Lee (2005). Minimizing total tardiness of orders with reentrant lots in a hybrid flowshop. International Journal of Production Research 43(11), 2149-2167.

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., K. Lee, K. Shin and J. Park (2005). A new approach to job shop scheduling problems with due date constraints considering operation subcontracts. International Journal of Production Economics 98(2), 238-250.

Coman, A. and B. Ronen (2000). Production outsourcing: a linear programming model for the theory-of-constraints. International Journal of Production Research 38(7), 1631-1639.

Feng, B., Z.-P. Fan and Y. Fi (2011). A decision method for supplier selection in multi-service outsourcing. International Journal of Production Economics 132(2), 240-250.

Hong, J. M. and I. S. Lee (2013). Integer programming approach for the outsourcing decision problem in a single machine scheduling problem with due date constraints. Korean Management Science Review 30(2), 133-141.

Kaipia, R. and K. Tanskanen (2003). Vendor managed category management an outsourcing solution in retailing. Journal of Purchasing and Supply Management 9(4), 165-175.

Kim, B. (2003). Dynamic outsourcing to contract manufacturers with different capabilities of reducing the supply cost. International Journal of Production Economics 86(1), 63-80.

Kolisch, R. (2000). Integration of assembly and fabrication for make-to-order production. International Journal of Production Economics 68(3), 287-306.

Koulamas, C. (1994). The total tardiness problem: review and extensions. Operations Research 42(6), 1025-1041.

Lee, Y. H., C. S. Jeong and C. Moon (2002). Advanced planning and scheduling with outsourcing in manufacturing supply chain. Computers & Industrial Engineering 43(1), 351-374.

Lee, I. S. and C. S. Sung (2008). Minimizing due date related measures for a single machine scheduling problem with outsourcing allowed. European Journal of Operational Research 186(3), 931-952.

Lee, I. S. and C. S. Sung (2008). Single machine scheduling with outsourcing allowed. International Journal of Production Economics 111(2), 623-634.

Lee, K. and B.-C. Choi (2011). Two-stage production scheduling with an outsourcing option. European Journal of Operational Research 213(3), 489-497.

Li, D.-F. and S.-P. Wan (2014). A fuzzy inhomogenous multiattribute group decision making approach to solve outsourcing provider selection problems. Knowledge-Based Systems 67(9), 71-89.

McCarthy, I. and A. Anagnostou (2004). The impact of outsourcing on the transaction costs and boundaries of manufacturing. International Journal of Production Economics 88(1), 61-71.

Mishra, N., A. K. Choudhary and M. K. Tiwari (2008). Modeling the planning and scheduling across the outsourcing supply chain: a chaos-based fast tabu-SA approach. International Journal of Production Research 46(13), 3683-3715.

Mokhtari, H., K. A. I. Nakhai 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(19), 5372-5395.

Neto, R. F. T. and M. G. Filho (2011). An ant colony optimization approach to a permutational flowshop scheduling problem with outsourcing allowed. Computers & Operations Research 38(9), 1286-1293.

Qi, X. (2011). Outsourcing and production scheduling for a two-stage flow shop. International Journal of Production Economics 129(1), 43-50.

Raman, N. and F. B. Talbot (1993). The job shop tardiness problem: A decomposition approach. European Journal of Operational Research 69(2), 187-199.