Industrial Engineering
& Management Systems
Vol 12, No 1, March 2013, pp.51-62
ISSN 1598-7248 | EISSN 2234-6473 |
http://dx.doi.org/10.7232/iems.2013.12.1.051

No Tardiness Rescheduling with Order Disruptions

Jaehwan Yang*
College of Business Administration, University of Seoul, Seoul, Korea

(Received: November 13, 2012 / Revised: January 14, 2013 / Accepted: February 18, 2013)

ABSTRACT

This paper considers a single machine rescheduling problem whose original (efficiency related) objective is minimizing makespan. We assume that disruptions such as order cancelations and newly arrived orders occur after the initial scheduling, and we reschedule this disrupted schedule with the objective of minimizing a disruption related objective while preserving the original objective. The disruption related objective measures the impact of the disruptions as difference of completion times in the remaining (uncanceled) jobs before and after the disruptions. The artificial due dates for the remaining jobs are set to completion times in the original schedule while newly arrived jobs do not have due dates. Then, the objective of the rescheduling is minimizing the maximum earliness without tardiness. In order to preserve the optimality of the original objective, we assume that no-idle time and no tardiness are allowed while rescheduling. We first define this new problem and prove that the general version of the problem is unary NP-complete. Then, we develop three simple but intuitive heuristics. For each of the three heuristics, we find a tight bound on the measure called *modified z-approximation ratio*. The best theoretical bound is found to be $0.5 - \varepsilon$ for some $\varepsilon > 0$, and it implies that the solution value of the best heuristic is at most around a half of the worst possible solution value. Finally, we empirically evaluate the heuristics and demonstrate that the two best heuristics perform much better than the other one.

Keywords: Rescheduling, Order Disruption, Heuristic Analysis, Minimizing Maximum Earliness

* Corresponding Author, E-mail: jyang@uos.ac.kr

1. INTRODUCTION

In most manufacturing facilities, disruptions occur constantly throughout their manufacturing processes. However, most of the scheduling studies focus on solving a scheduling problem without assuming any disruptions after an initial schedule is developed. A disruption is defined as a state where the deviation from plan is sufficiently large that the current plan has to be changed substantially (Clausen *et al.*, 2001). Examples of common disruptions include the arrival of new orders, order cancelations, changes in order priority, processing delays, changes in release dates, machine breakdowns, and the unavailability of raw materials, personnel, or tools (Hall and Potts, 2004).

If disruptions occur in a manufacturing facility, the operation manager (OM) needs to update or reschedule the current schedule by considering the type and magnitude of the disruptions. While rescheduling, it is possible to ignore the current schedule and develop a completely new schedule. However, it will cause other problems with respect to reallocations of raw materials and resources including labor, tools and equipment, which had already been prepared for the current schedule. Furthermore, the OM has to deal with the unhappy sales department or customers thanks to substantial changes in completion time of some orders. Hence, it would be necessary to consider the trade-off between the effici-ency related cost of scheduling and the disruption related cost of scheduling when the new schedule is generated.

In addition to the general necessity of rescheduling in the real world, a direct motivation of the paper can be found at the manufacturing facility of a global consumer electronics company. In this facility, orders are scheduled with a traditional objective such as minimizing makespan or minimizing total completion time. After an initial schedule is generated, each order has an estimated completion time or so called quoted due date (QDD), which is used to notify customers when their order would be shipped from the factory. Then, whenever rescheduling is necessary due to the order disruptions, these quoted completion times are often hard-pegged as fixed completion times. Obviously, the company does not want to see the changes in the already determined completion times, and thus they can minimize the effect of the disruptions.

Specifically, we consider a single machine case where the original (efficiency related) objective is minimizing makespan, and the disruption related objective is minimizing maximum earliness without tardiness. This type of situation can be found in many industries where no particular priority is given to each of the orders. The due dates for the remaining jobs are set to completion times in an original schedule, and new jobs do not have any due dates. In order to preserve the optimality of the original objective, no-idle time is allowed while rescheduling. In addition, we assume that new jobs are available to be processed at the beginning of planning horizon, no tardiness is allowed for the remaining jobs, and no preemption is allowed for all jobs.

For general discussions on rescheduling, see Hall and Potts (2004). For practical significance on this type of work, see Clausen *et al.* (2001) and Kopanos *et al.* (2008). There are several studies which are closely related to the paper. Wu *et al.* (1993) consider a single machine rescheduling problem where job ready times and tails exist. This problem is easily related to a job shop environment, and machine failure is considered as disruption. The objective is to minimize the makespan and the disruption from the original schedule. The disruption is measured with the objective of the minimization of total earliness and tardiness and the start times of jobs in the original schedule are used as dues dates. Unal *et al.* (1997) study a single machine with newly arrived jobs that have setup times that depend on their part types. One objective is to minimize the total weighted completion time, and the other is to minimize the makespan of the new schedule. While not considering a specific objective function to measure the disruption, the constraints such as fixed sequence for remaining jobs and no additional setups are used to reduce the impact of the disruption. They provide efficient algorithms, complexity results, and heuristics with guaranteed performance bounds. Hall and Potts (2004) also consider scheduling a single machine problem with newly arrived jobs as disruptions. Two separate objective functions such as minimizing position difference between starting times in the original schedule and the new schedule and minimizing total earliness and tardiness as the start times of jobs in the original schedule as due dates are used to minimize disruptions. They provide several intractability results and heuristics with the performance evaluation.

Qi *et al.* (2006) study a single machine and two parallel machine rescheduling problems where unexpected machine breakdown or changes in processing time occur. The efficiency related objective is to minimize the total completion time, and the disruption related objective is to minimize total earliness and tardiness as the start times of jobs in the original schedule as due dates. A composite objective function is used to address the both types of objectives. Ozlen and Azizoglu (2011) consider rescheduling in an unrelated parallel shop where unexpected machine failure occurs. Their efficiency related objective is to minimize the total completion time, and the disruption related objective is to minimize the number of jobs which are assigned to a different machine or to minimize the cost associated with the number of jobs that are assigned to a different machine. They provide polynomial-time solution methods to some hierarchical optimization problems and propose exponential time algorithms to generate all efficient solutions and to minimize a specified function of the measures.

Finally, Yang and Posner (2012) consider a single machine rescheduling problem where the objective is to minimize the maximum deviation where both tardiness and earliness are allowed. They establish the complexity of the problem and develop a few heuristics. They also find that their results can be easily extended to a special case of the problem with the objective of minimizing total earliness and tardiness.

While this work is similar to Yang and Posner (2012), it is focused on the rescheduling situation where tardiness is not allowed. This situation can be found in the real world when the initially determined completion time is used to enforce as a promised delivery date to customers and thus, it becomes a deadline which must be met all the time. However, the rescheduling situation considered in Yang and Posner (2012) allows both tardiness and earliness, and tardiness and earliness are equally penalized when rescheduling is performed.

Theoretically, these two problems are differentiated from each other mainly due to different objective functions and the existence of no-tardy constraint in our problem, and hence need different heuristics to obtain efficient schedules. For instance, one of the heuristics in Yang and Posner (2012) has a parameter which controls how much tardiness is allowed compared to the size of the immediately preceding gap. Depending upon the size of the parameter, the empirically evaluated performance of the heuristic varies significantly. Another heuristic includes a step which determines whether some new job which creates tardiness should be scheduled at a specific gap based on a local optimal rule. Furthermore, their theoretical analysis such as proofs of the worst case bound of the heuristics is completely different from those found in this paper since different problems and heuristics are considered. Nonetheless, we follow the

approaches similar to those used in Yang and Posner (2012) when we analyze our problem due to their similarity in structure of the two problems.

This paper considers an objective function that is the minimization of maximum earliness. Several studies consider this objective in a regular scheduling environment (Azizoglu *et al.*, 2003; Guner *et al.*, 1998; Mandel and Mosheiov, 2001; Molaee *et al.*, 2010). However, few have considered this objective function in a disruption related rescheduling problem. Moreover, this work is differentiated from other works which consider the order disruption because we consider both newly arrived and canceled jobs while the other studies consider only newly arrived orders except for Yang and Posner (2012).

In the next section, we introduce some notation and describe the problem. In Section 3, we present preliminary results. Then, we prove the complexity of the problem and present solutions for special cases in Section 4. Four heuristics are introduced in Section 5. We first introduce a heuristic which always generates the maximum value of earliness and is used as an upper bound for all the other three heuristics. Then, we introduce three heuristics that are based on scheduling jobs either in the index order or in the longest processing time order at the first available idle times created by cancelled jobs. In Section 6, we suggest a new evaluation measure called *modified z-approximation ratio* for the heuristics. Then, for each of the three heuristics, we find a tight bound on the modified z-approximation ratio in Section 7. Finally, we empirically evaluate the heuristics in Section 8.

2. NOTATION

The parameters of the problem are

n_o : number of original jobs
n_c : number of cancelled jobs
$n_r = n_o - n_c$: number of remaining jobs
n_n : number of new jobs
$n = n_r - n_n$: number of jobs to be scheduled
N_r : set of remaining jobs $= \{1, 2, \cdots, n_r\}$
N_n : set of new jobs $= \{n_r + 1, n_r + 2, \cdots, n_r + n_n\}$
$N = N_r \cup N_n = \{1, 2, \cdots, n\}$
p_j : processing time of job j for $j \in N$
σ^d : disrupted schedule.

The decision variables are

σ : schedule of all jobs
$C_j(\sigma)$: completion time of job j in schedule σ for $j \in N$
$E_j(\sigma)$: earliness cost of remaining job j in schedule σ for $j \in N_r$
$E_{max}(\sigma) = \max_{j \in N_r}\{E_j(\sigma)\}$: maximum earliness in schedule σ for $j \in N_r$
$z(\sigma)$: value of schedule σ
σ^* : an optimal schedule
z^* : value of optimal schedule σ^*.

When no confusion exists, we replace $C_j(\sigma)$, $E_j(\sigma)$, and $E_{max}(\sigma)$ with C_j, E_j, and E_{max}, respectively. 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 and restrictions, and α_3 defines the objective. Following the standard scheduling classification schedule of Graham *et al.* (1979), we refer to the problem of minimizing maximum earliness cost in one machine as $1 | disrupt, no-idle, no-tardy | E_{max}$ where *disrupt* implies that the problem considers disruptions related rescheduling, *no-idle* means there exists no-idle time in a schedule, and *no-tardy* indicates that tardiness is not allowed for the remaining jobs. Note that only the remaining jobs have due dates which are set to the completion times in the original schedule. We assume that the remaining and new jobs are available at the start of the planning process. Also, preemptions are not allowed.

In this paper, a *schedule* defines a job order on the machine. Since no-idle time is allowed, the job order determines the start and completion time of jobs on the machine. An *original schedule* is the schedule which is created and established before disruptions occur. As we noted, the execution of the original schedule is disrupted because of cancelled and newly arrived jobs. Thus, there is a rescheduling. A *disrupted schedule* is the original schedule with canceled jobs removed. We assume that the disrupted schedule starts at the beginning of time horizon.

For notational convenience, each block of idle times created by canceled jobs is called *gap*. Let G be a set of gaps such that $G = \{G_1, G_2, \cdots, G_q\}$ where q is the number of gaps in disrupted schedule σ^d. Then, σ^d can be described as a sequence of jobs and gaps. For instance, $\sigma^d = (1, G_1, 2, G_2, 3, 4)$ means that there are four remaining jobs 1, 2, 3, and 4, and one gap between jobs 1 and 2 and another gap between jobs 2 and 3. We also, let g_j be duration of gap G_j for $j \in \{1, 2, \cdots, q\}$. Note that $g_j > 0$ for all $j \in \{1, 2, \cdots, q\}$. Also, we let R_i be a set of remaining jobs processed between gaps i and $i+1$ for $i = 0, 1, \cdots, q-1$. We also let r_i be the number of jobs in R_i for $i = 0, 1, \cdots, q$. Then, $r_0 \geq 0$ and $r_i > 0$ for $i = 1, 2, \cdots, q$.

3. PRELIMINARY RESULTS

We first develop some results that provide the basis for our analysis. Then we establish the complexity of the problem. First, we prove the following two lemmas which establish relationships between σ^d and σ^*. For notational convenience, we let $d_j = C_j(\sigma^d)$ for $j = 1, 2, \cdots, n_r$ and d_j is considered as due date in the rescheduling problem.

Lemma 1. There exists an optimal schedule where the order of the remaining jobs is the same as that in σ^d.

Proof. Suppose there exist adjacent remaining jobs i and $j \in N_r$ such that job i precedes job j in σ^d but job j precedes job i in σ^*. Notice that some new jobs in N_n can be scheduled between these two remaining jobs in σ^*. Since job j precedes job i in σ^* and $d_i < d_j$, $E_i(\sigma^*) < E_j(\sigma^*)$. If we switch jobs i and j in σ^*, then the both jobs still incur earliness cost and further, earliness cost of job i increases but earliness cost of job j decreases. Since $d_i < d_j$, the new earliness cost of job i is less than $E_j(\sigma^*)$. Also, the new earliness cost of job j is less than $E_j(\sigma^*)$.

Therefore, changing the order of the remaining jobs generates a better schedule, and thus σ^* is not optimal. Contradiction. □

As a result of Lemma 1, we only consider a schedule where the order of remaining jobs is the same as in σ^d.

Lemma 2. There exists an optimal schedule where the two remaining jobs that process consecutively in σ^d still process consecutively in σ^*.

Proof. Suppose that there exists an optimal schedule where some new jobs are scheduled between two consecutively processing remaining jobs in σ^d. Suppose that $r \in N_n$ is the first such job and job r is scheduled between two remaining jobs i and j for $i, j \in N_r$ in σ^*. Notice that earliness cost cannot incur on job j only because jobs i and j are scheduled consecutively. If earliness cost incurs either on both jobs i and j or job i only in σ^*, then we can reduce the earliness cost by scheduling job r before job i. This contradicts that the schedule is optimal. We use the same argument for all new jobs which are scheduled between two consecutively processing remaining jobs in σ^d. □

As a result of Lemma 2, when we consider an optimal schedule, we assume that no new jobs are scheduled between two consecutively processing remaining jobs in σ^d.

4. COMPLEXITY RESULTS

The next result establishes the complexity of the problem. We show that problem $1|disrupt, no-idle, no-tardy|E_{max}$ is unary NP-complete by using the reduction from 3-Partition which is a known unary NP-complete problem.

3-Partition (Garey and Johnson, 1979). Given 3ℓ elements with integer sizes $a_1, a_2, \cdots, a_{3\ell}$, where $\sum_{j=1}^{3\ell} a_j = \ell y$ and $y/4 < a_j < y/2$ for $j = 1, 2, \cdots, 3\ell$, does there exist a partition $A_1, A_2, \cdots, A_\ell$ of the index set $\{1, 2, \cdots, 3\ell\}$ such that $|A_i| = 3$ and $\sum_{j \in A_i} a_j = y$ for $i = 1, 2, \cdots, \ell$?

Theorem 1. Problem $1|disrupt, no-idle, no-tardy|E_{max}$ is unary NP-complete.

Proof. Given an instance of 3-Partition, we construct the following instance of $1|disrupt, no-idle, no-tardy|E_{max}$:

$$
\begin{aligned}
&n_r = \ell,\\
&n_n = 3\ell,\\
&p_j = 1,\ j = 1, 2, \cdots, \ell,\\
&p_j = a_j,= j = \ell+1, \ell+2, \cdots, 4\ell,\\
&g_i = y,\ i = 1, 2, \cdots, \ell,\\
&\sigma^d = (G_1, 1, G_2, 2, \cdots, G_\ell, \ell),\\
&z = 0.
\end{aligned}
$$

The decision version of problem $1|disrupt, no-idle, no-tardy|E_{max}$ is in NP since we can calculate E_{max} in polynomial time. We assume without loss of generality that $\sum_{j=1}^{3\ell} a_j = \ell y$. We prove that there exists a solution to 3-Partition if and only if there exists a solution to problem $1|disrupt, no-idle, no-tardy|E_{max}$ where $E_{max} = 0$.

($\Rightarrow$) Suppose that there exists a solution to 3-Partition. Then, there exist ℓ disjoint set $A_1, A_2, \cdots, A_\ell$ such that $\sum_{a_j \in A_i} a_j = y$ for $i = 1, 2, \cdots, \ell$. We assume without loss of generality that, if there exists a solution to 3-Partition, then the elements are indexed such that $a_{3i-2} + a_{3i-1} + a_{3i} = y$ for $i = 1, 2, \cdots, \ell$. Consider schedule σ for problem $1|disrupt, no-idle, no-tardy|E_{max}$ where

$$
\begin{aligned}
\sigma = (&\ell+1, \ell+2, \ell+3, 1, \ell+4, \ell+5, \ell+6,\\
&2, \cdots, 4\ell-2, 4\ell-1, 4\ell, \ell).
\end{aligned} \tag{1}
$$

Note that $p_{\ell+3i-2} + p_{\ell+3i-1} + p_{\ell+3i} = y$ for $i = 1, 2, \cdots, \ell$. Thus, jobs $\ell+3i-2$, $\ell+3i-1$, and $\ell+3i$ are scheduled at G_i without generating earliness cost on job i for $i = 1, 2, \cdots, \ell$. Therefore, $E_{max} = 0$ for problem $1|disrupt, no-idle, no-tardy|E_{max}$.

($\Leftarrow$) Suppose that there exists a solution to problem $1|disrupt, no-idle, no-tardy|E_{max}$ where $E_{max} = 0$. Since $E_{max} = 0$, new jobs should be scheduled at each gap without creating any idle time. Furthermore, new jobs should not generate any tardiness. This implies that the total processing time of new jobs which are scheduled at each gap must be exactly y. Observe that $y/4 < p_j < y/2$ for $j = \ell+1, \ell+2, \cdots, 4\ell$. Also, there are ℓ gaps and 3ℓ new jobs. Hence, solving the problem with $E_{max} = 0$ is possible only if a set of three new jobs are scheduled at each gap and the sum of the processing times of each set of three jobs is exactly y. This is only possible if there exists a solution to 3-Partition. □

The next result establishes the complexity of a special case of problem $1|disrupt, no-idle, no-tardy|E_{max}$ where there exists fixed number of gaps in a disrupted schedule. We use the reduction from the following binary NP-complete problem (Karp, 1972).

Partition. Given a multiset A of $2n$ integers and a positive integer b, can set $A = \{a_1, a_2, \cdots, a_{2n}\}$ be partitioned into two disjoint subsets A_1 and A_2 such that the sum of the elements in A_1 equals the sum of the elements in A_2 such that $\sum_{a_j \in A_1} a_j = \sum_{a_j \in A_2} a_j = b$?

Lemma 3. If there exists only one gap in σ^d and $g_1 < \sum_{j \in N_n} p_j$, then the recognition version of the problem is at least binary NP-complete.

Proof. Given an instance of Partition problem, we construct the following instance of problem $1|disrupt, no-idle, no-tardy|E_{max}$ with one gap and $g_1 < \sum_{j \in N_n} p_j$. We assume that there exist one remaining job, job 1 and $2n$ new jobs, and let $p_1 = 1$ and $p_j = a_j$ for $j \in N_n$. We also set g_1 to the size of a partition such that $g_1 = b$, and $\sigma^d = (G_1, 1)$.

Notice that the decision version of problem $1|disrupt, no-idle, no-tardy|E_{max}$ with one gap and $g_1 < \sum_{j \in N_n} p_j$ is in NP since we can calculate E_{max} in polynomial time. We prove that there exists a solution to Partition if and only if there exists a solution to problem $1|disrupt, no-idle, no-tardy|E_{max}$ where there exists one gap and $g_1 < \sum_{j \in N_n} p_j$ with $E_{max} = 0$.

First, having a partition with the size of b implies that there exists a set of jobs in N_n such that their total processing time is equal to b which is also equal to g_1. Then, we schedule those jobs at G_1 and the rest of jobs in N_n at the end of σ^d. Since the scheduled new jobs at the gap do not create any earliness cost on job 1, the solution value of this schedule is 0.

In the other direction, having a schedule for the problem with $E_{max} = 0$ implies that there exist a set of jobs in N_n such that their total processing time is equal to b. Since $p_j = a_j$ for $j \in N_n$, there exists a partition. Therefore, the partition is possible only if there exists an optimal schedule for problem $1|disrupt, no-idle, no-tardy|E_{max}$ where there exists one gap and $g_1 < \sum_{j \in N_n} p_j$ with $E_{max} = 0$. □

The following result establishes that there exists an optimal procedure which runs in pseudo-polynomial time for problem $1|disrupt, no-idle, no-tardy|E_{max}$ with a fixed number of gaps.

Lemma 4. If there exists a fixed number of gaps in σ^d, then there exists an optimal procedure which runs in pseudo-polynomial time.

Proof. Notice that the recognition version of the bin packing problem with a fixed number of bins is binary NP-complete, and there exists an optimal solution procedure which runs in pseudo-polynomial time (Garey and Johson, 1979). In addition, this result can be extended to the case where the size of bins is variable. This result implies that for the problem with a fixed number of gaps, we can recognize whether new jobs can be scheduled at gaps without changing starting times of the remaining jobs and without idle time in pseudo-polynomial time.

Now, let $s = \sum_{k=1}^{q} g_k$ where q is the number of gaps. Observe that while solving our problem with a fixed number of gaps, the maximum possible size of any gap is s. Since the number of gaps is fixed, the possible number of problems with different combinations of gap sizes is at most $O(s^q)$. This is due to the fact that the sum of sizes of all gaps is always no greater than s. Moreover, as described, we can check whether new jobs can be scheduled at gaps without generating earliness cost for each of these possible problems in pseudo-polynomial time. Then, by comparing each of the results, we can find a schedule that minimizes the maximum earliness cost for the problem with a fixed number of gaps in pseudo-polynomial time. Therefore, there exists an optimal procedure which runs in pseudo-polynomial time for problem $1|disrupt, no-idle, no-tardy|E_{max}$ with a fixed number of gaps in σ^d. □

Theorem 2. Problem $1|disrupt, no-idle, no-tardy|E_{max}$ with a fixed number of gaps in σ^d is binary NP-complete.

Proof. The result follows from Lemmas 3 and 4. □

5. HEURISTICS

The heuristics in this section are developed based on one of the simple and intuitive heuristics for the bin-packing problem which is similar to our problem in a way that items (jobs) should be packed (scheduled) at multiple bins (gaps) as much as possible. They are first-fit (FF) and first-fit decreasing (FFD) heuristics developed and analyzed in Johnson *et al.* (1974). The heuristic FF assigns items to bins according to the order they appear in the list without using any knowledge of subsequent items in the list. Meanwhile, the FFD first sorts the items in nonincreasing order of their size and then assigns items to bins according to the order they appear in the list. In our problem, the FF schedules new jobs in their index order at gaps according to the sequence of the gaps in which they appear in the disrupted schedule. The FFD is similar to the FF except that the FFD first sorts the new jobs in nonincreasing order of their size before it schedules them at each of the gaps. Using the longest processing time job first (LPT) rule for scheduling new jobs at each gap is intuitive and supposed to be more effective in reducing the earliness cost incurred at gaps than the list scheduling (LS) rule where jobs are

scheduled in their index order. Now, we formally describe each of the heuristics.

Heuristic H0 generates a schedule with maximum earliness cost for each remaining job. It simply processes all remaining jobs without idle time before any new job. Then, all new jobs are scheduled at the end of σ^d. It also provides a good upper bound for the analysis of the other heuristics.

Heuristic H0
1. Process all remaining jobs without idle time before any new job.
2. All new jobs are scheduled at the end of the disrupted schedule.

Heuristic H1 uses the LS rule to select a new job to be scheduled at each gap, and uses the index order to select the next gap to be considered.

Heuristic H1
0. Set $i = 0$.
1. If $G = \varnothing$ or $N_n = \varnothing$, then go to step 5.
2. Set $r = |N_n|$, $j = n_r$, and $i = i+1$.
   Reindex new jobs in N_n from $n_r + 1$ to $n_r + r$ without changing the order of jobs.
3. Set $j = j + 1$.
   If $p_j \le g_i$, then schedule job j at G_i, $g_i = g_i - p_j$, and $N_n = N_n \setminus \{j\}$.
4. If job j+1 exists in N_n and $g_i > 0$, then go to step 3. Otherwise, schedule jobs in R_i without idle time, $G = G \setminus \{G_i\}$, $g_{i+1} = g_{i+1} + g_i$, and go to step 1.
5. Eliminate remaining gaps by expediting jobs.
   All unscheduled new jobs are scheduled at the end of the current schedule.
   Calculate solution cost and stop.

Next, we introduce heuristic H2 which is identical to heuristic H1 except for the fact that it uses the LPT rule instead of the LS rule to select a new job to be scheduled first.

Heuristic H2
1. Follow H1, but use the LPT rule instead of the LS rule.

Heuristic H3 is similar to H2 except that it effectively handles the situation where a job bigger than the size of the gap at which it is scheduled is a part of an optimal schedule. For instance, suppose that $g_1 = 1$, $g_2 = 10$, $r_1 = 1$, $r_2 = 1$, and $\sigma^d = (G_1, 1, G_2, 2)$. Also, let $p_1 = p_2 = 1$, $p_3 = \varepsilon$, and $p_4 = 11$ and jobs 3 and 4 are new jobs for some small $\varepsilon > 0$. Notice that $\sigma^{H2} = (3, 1, 2, 4)$ and $z^{H2} = 11 - \varepsilon$. On the other hand, $\sigma^* = (1, 4, 2, 3)$ and $z^* = 1$. This situation occurs because σ^* includes a big job which is bigger than the original size of G_2 at which it is scheduled. Neither H1 nor H2 can generate a reasonably good schedule since they only consider each gap at a time.

The following heuristic handles this situation by trying to schedule a big new job when it recognizes a high earliness cost at a gap. Specifically, if the earliness cost is bigger than a half of the total size of gaps so far, it tries to schedule a big new job which can be scheduled at the gap by unscheduling some new jobs at previous gaps. In the example above, H3 would schedule job 4 when the size of G_2 is $11 - \varepsilon$ after job 3 is scheduled at G_1, and eventually, it would generate the same schedule as σ^*. For notational convenience, let P_i be the total processing time of new jobs scheduled at G_i and all earlier gaps for $i = 1, 2, \cdots, q$. We now formally describe heuristic H3 as follows.

Heuristic H3
0. Set $g'_i = \bar{g}_i = g_i$ for $i = 1, 2, \cdots, q$.
1. If $G = \varnothing$ or $N_n = \varnothing$, then go to step 6.
2. Set $r = |N_n|$, and $j = n_r$.
   Let i be the index of the first available gap.
   Reindex new jobs in N_n so that $p_j \ge p_{j+1}$ for $j = n_r + 1, n_r + 2, \cdots, n_r + r - 1$.
3. Set $j = j + 1$.
   If $p_j \le g'_i$, then schedule job j at G_i, $g'_i = g'_i - p_j$, and $N_n = N_n \setminus \{j\}$.
4. If job $j+1$ exists in N_n and $g'_i > 0$, then go to step 3.
5. If $g'_i \ge \sum_{k=1}^{i} g_k/2$ and there exists new job s such that $g'_i < p_s \le \sum_{k=1}^{i-1} g_k + \bar{g}'_i$, then perform the following three substeps.
5.1 Set job $\bar{s} = \text{argmin}\{p_s \mid g'_i < p_s \le \sum_{k=1}^{i-1} g_k + \bar{g}'_i\}$ and $\bar{u} = \min\{u \mid p_{\bar{s}} \le \sum_{k=1}^{u} g_{i-k} + \bar{g}'_i\}$.
5.2 Unschedule all new jobs at $G_{i-\bar{u}}, G_{i-\bar{u}+1}, \cdots, G_i$, include them back in N_n, and schedule job $\bar{s}$ at G_i as late as possible without tardiness.
5.3 Set $g'_{i-\bar{u}} = \bar{g}'_{i-\bar{u}} = \sum_{k=1}^{i-1} g_k + \bar{g}'_i - (p_{\bar{s}} + P_{i-\bar{u}-1})$, $N_n = N_n \setminus \{\bar{s}\}$, $G = G \cup \{G_{i-\bar{u}}\}$, and go to step 1.
6. Schedule jobs in $R_i, R_{i+1}, \cdots, R_{\ell-1}$ without idle time, set $G = G \setminus \{G_i\}$, and $g'_\ell = g_\ell + g'_i$ where ℓ is the index of the next available gap. Go to step 1.
7. Eliminate remaining gaps by scheduling jobs without idle time.
   All unscheduled new jobs are scheduled at the end of the current schedule.
   Calculate solution cost and stop.

In step 5, $\bar{g}_i$ is identical to g_i unless job $\bar{s}$ is already scheduled at G_{i+1} and thus, jobs in R_i are expedited. In step 5.2, jobs in $R_{i-\bar{u}+1}, R_{i-\bar{u}+2}, \cdots, R_{i-1}$ would be expedited when job $\bar{s}$ is scheduled at G_i, and job $\bar{s}$ is scheduled at G_i such that earliness cost of job $\bar{s}$ is zero.

6. EVALUATION MEASURE FOR HEURISTICS

The problem being considered has the objective of minimizing E_{max}, and it is possible that $z^* = 0$. Hence, it is unsuitable to use relative error measure $(z^H - z^*)/z^*$ to analyze the heuristics. As an alternative, we consider the *z-approximation ratio* by Hassin and Khuller (2001). An

algorithm is an α z-approximation if it runs in polynomial time and produces a solution whose distance from the optimal one is at most α times the distance between the optimal solution and the worst possible solution. Then, the goal is to find a schedule σ with the property that

$$zr = \frac{z(\sigma) - z*}{z(\sigma^w) - z^*} \le \alpha, \quad (2)$$

where $0 \le \alpha \le 1$ and σ^w is the worst schedule which generates a maximum value of earliness.

However, the regular z-approximation ratio described above may have $z(\sigma^w) = z^*$. For instance, suppose $\sigma^d = (G_1, 1)$ and $g_1 = 1$ and $p_1 = 1$. Also, there exists a new job, job 2 such that $p_2 = 2$. Schedule (1, 2) can be an optimal and worst case schedule. Then, the denominator of the z-approximation ratio becomes zero.

Also, for the same gaps and the same remaining jobs, σ^w may have a sequence of remaining jobs different from that in σ^d. As in Lemma 1, there exists an optimal schedule where the order of remaining jobs is fixed. Hence, not all the heuristics attempt to change the order of the remaining jobs. However, $z(\sigma^w)$ would have a different order of remaining jobs, and this makes $z(\sigma^w)$ too large and artificial to be used in evaluating heuristics.

Notice that the first concern is a technical issue which may be overcome by specifying a value for the denominator when this situation occurs. However, the other issue cannot be resolved with the current form of the z-approximation in (2). Specifically, the worst solution value can become too large simply by changing the order of the remaining jobs, which is fixed for all the heuristics being evaluated.

Hence, we need a new heuristic which can replace the worst schedule solution in (2). This heuristic should consistently generate a *worse* solution value than heuristics being evaluated while changing similarly to optimal and most heuristic solution values as the structure of the problem changes. Moreover, its solution value should be easily computed with the least calculation effort. The following lemma establishes that heuristic H0 generates the worst case solution value if the remaining job sequence is fixed as in σ^d.

Lemma 5. For problem $1 \mid disrupt,\ no-idle, no-tardy \mid E_{max}$ with the fixed remaining job sequence, heuristic H0 always generates the worst case solution value.

Proof. Heuristic H0 schedules each job in R_i for i = 1, 2, ⋯, q at the earliest possible time. □

As a result of Lemma 5, we replace $z(\sigma^w)$ by z^{H0} in (2). No heuristic which keeps the same remaining job sequence as in σ^d should be worse than heuristic H0. Hence, we have the following formal definition for the modified z-approximation ratio.

$$zr' = \begin{cases} 0 & if\ z(\sigma^{H0}) = z^* = z(\sigma) \\ \infty & if\ z(\sigma^{H0}) = z^*\ and\ z(\sigma) \ne z(\sigma^{H0}) \\ \dfrac{z(\sigma) - z(\sigma^*)}{z(\sigma^{H0}) - z^*} if & z(\sigma^{H0}) \ne z^* \end{cases}$$

where σ^{H0} is a schedule by H0 and σ is a schedule by the heuristic being evaluated. Notice that $zr' \le 1$. Check-ing whether $z(\sigma^{H0}) = z^*$ can be easily done as follows. If $\sum_{i=1}^{q} g_i > \min_{j \in N_n}\{p_j\}$, then $z(\sigma^{H0}) > z^*$. Otherwise, $z(\sigma^{H0}) = z^*$.

7. WORST CASE BOUND ON MODIFIED Z-APPROXIMATION

7.1 Heuristic H1

The following result establishes the tight worst case bound on the modified z-approximation for H1.

Theorem 3. For problem $1 \mid disrupt, no-idle, no-tardy \mid E_{max}$, $(z^{H1} - z^*)/(z^{H0} - z^*) \le 1 - \varepsilon$ for some $\varepsilon > 0$ and this bound is tight.

Proof. If an optimal schedule has at least one job scheduled at any gap, then H1 also schedules at least one job at some gap. Hence, zr' cannot be equal to 1 and should be less than 1. Thus, we establish the result by presenting an instance where zr' can be as close to 1 as possible by controlling ε for some $\varepsilon > 0$.

Consider an instance of problem $1 \mid disrupt,\ no-idle, no-tardy \mid E_{max}$ where $N_r = \{1\}$, $N_n = \{2, 3\}$, and $\sigma^d = (G_1, 1)$. Also, $g_1 = 1$, $p_1 = 1$, $p_2 = \varepsilon$, and $p_3 = 1$ for some small $\varepsilon > 0$. Since H1 uses the LS to schedule a job, the first job available, job 2 is scheduled at G_1 in σ^{H1}. Hence, $\sigma^{H1} = (2, 1, 3)$ with the solution value of $z^{H1} = 1 - \varepsilon$. However, optimal schedule $\sigma^* = (3, 1, 2)$ with the solution value of $z^* = 0$. For this instance, $\sigma^{H0} = (1, 2, 3)$ with the solution value of $z^{H0} = 1$. Therefore, $zr' = (1 - \varepsilon)/1 = 1 - \varepsilon \approx 1$. Notice that zr' can be as close to 1 as we want by reducing ε. Hence, it should be the worst case example of H1. Also, since there exists an instance of the problem, the bound is tight. □

7.2 Heuristic H2

The following result establishes the tight worst case bound on the modified z-approximation for H2.

Theorem 4. For problem $1 \mid disrupt, no-idle, no-tardy \mid E_{max}$, $(z^{H2} - z^*)/(z^{H0} - z^*) \le 1 - \varepsilon$ for some $\varepsilon > 0$ and this bound is tight.

Proof. If an optimal schedule has at least one job scheduled at any gap, then heuristic H2 also schedules at

least one job at some gap. Hence, zr' cannot be equal to 1 and should be less than 1. Thus, we establish the result by presenting an instance where zr' can be as close to 1 as possible by controlling ε for some $\varepsilon > 0$.

Consider an instance of problem $1|disrupt, no-idle, no-tardy|E_{max}$ where $N_r = \{1, 2\}$, $N_n = \{3, 4\}$, and $\sigma^d = (G_1, 1, G_2, 2)$. Also, $g_1 = g_2 = 1$, $p_1 = p_2 = 1$, $p_3 = \varepsilon$, and $p_4 = 2$ for some small $\varepsilon > 0$. Since $p_4 > g_1$, job 3 should be scheduled first in σ^{H2}, then, job 4 is considered for G_2 but scheduling job 4 would generate a schedule with tardiness. Hence, $\sigma^{H2} = (3, 1, 2, 4)$ with the solution value of $z^{H2} = 2 - \varepsilon$. However, optimal schedule $\sigma^* = (1, 4, 2, 3)$ with the solution value of $z^* = 1$. For this instance, $\sigma^{H0} = (1, 2, 3, 4)$ with the solution value of $z^{H0} = 2$ and $zr' = (2 - \varepsilon - 1)/(2 - 1) = 1 - \varepsilon \approx 1$. Notice that zr' can be as close to 1 as we want by reducing ε. Hence, it is the worst case example of H2. Also, since there exists an instance of the problem, the bound is tight. □

7.3 Heuristic H3

We begin by analyzing H3 for problem $1|disrupt, no-idle, no-tardy|E_{max}$ with one gap. Then, we extend our analysis to a general case. First, the following example presents a possible worst case instance on the modified z-approximation ratio for H3.

Example 1. Consider an instance of problem $1|disrupt, no-idle, no-tardy|E_{max}$ where $N_r = \{1\}$, $N_n = \{2, 3, 4\}$, and $\sigma^d = (G_1, 1)$. Also, $g_1 = 1$, $p_1 = 1$, $p_2 = 0.5 + \varepsilon$, and $p_3 = p_4 = 0.5$ for some small $\varepsilon > 0$. Since job 2 is the biggest among the new jobs, job 2 should be scheduled first in σ^{H3}. Then, jobs 3 and 4 cannot be scheduled at the gap because they are bigger than the remaining gap. Hence, $\sigma^{H3} = (2, 1, 3, 4)$ with the solution value of $z^{H3} = 0.5 - \varepsilon$. However, optimal schedule $\sigma^* = (3, 4, 1, 2)$ with the solution value of $z^* = 0$. For this instance, $\sigma^{H0} = (1, 2, 3, 4)$ with the solution value of $z^{H0} = 1$ and $zr' = (0.5-\varepsilon)/0.5-\varepsilon$. □

The following result shows that if there exists only one gap in σ^d, then the worst case bound on the modified z-approximation for H3 is $0.5-\varepsilon$ for some small $\varepsilon > 0$. Notice that if there exists only one gap, then H3 works the same way as H2.

Lemma 6. If there exists only one gap in σ^d, then $zr' = (z^{H3} - z^*)/(z^{H0} - z^*) \le 0.5 - \varepsilon$ for some small $\varepsilon > 0$ and this bound is tight.

Proof. Without loss of generality, we assume that $g_1 = 1$ and $N_r = \{1\}$ with $p_1 = 1$. For notational convenience, let P_1^{H3} and P_1^* be the total processing time of new jobs which are scheduled at G_1 in σ^{H3} and σ^*, respectively. If no job is scheduled at G_1 in σ^*, then $z^* = z^{H0}$, and thus, we do not consider this case. Suppose that only one job is scheduled at G_1 in σ^*. Observe that H3 uses the LPT to select the first job to be scheduled at G_1. Hence, σ^* and σ^{H3} should be identical.

Now, suppose that more than one jobs are scheduled at G_1 in σ^*. Notice that in the worst case, the first job in σ^* and σ^{H3} should be different, and also the first job in σ^{H3} should be bigger than that in σ^* to have two different schedules. From the definition of the modified z-approximation ratio,

$$zr' = \frac{E_1(\sigma^{H3}) - E_1(\sigma^*)}{E_1(\sigma^{H0}) - E_1(\sigma^*)} = \frac{(1 - P_1^{H3}) - (1 - P_1^*)}{1 - (1 - P_1^*)} = \frac{P_1^* - P_1^{H3}}{P_1^*} = 1 - \frac{P_1^{H3}}{P_1^*}. \quad (3)$$

In order to have $zr' \ge 0.5$, P_1^{H3} should be no greater than 0.5. If $P_1^{H3} = 0.5$, then $P_1^* = 1$ to have $zr' = 0.5$. However, this case is impossible because one of the new jobs scheduled at G_1 in σ^* should be no bigger than 0.5 to have $P_1^* = 1$. Contradiction. Hence, $P_1^{H3} < 0.5$ to have $zr' \ge 0.5$.

Without loss of generality, we suppose that jobs are sequenced in a nonincreasing order at G_1 in σ^*. If only two new jobs are scheduled at G_1 in σ^*, then the second job must be scheduled in σ^{H3}, and due to the LPT rule, σ^{H3} should be better than σ^*. Contradiction. Alternatively, if there exist more than two new jobs that are scheduled at G_1 in σ^*, then the size of the second job is less than 0.5, the size of the third job is no greater than 1/3, and so on. Hence, if $P_1^* \ge 0.5$, then P_1^{H3} should be no less than 0.5 due to the LPT rule and the availability of new jobs scheduled at G_1 in σ^*. Alternatively, if $P_1^* < 0.5$, then it should be that $P_1^{H3} = P_1^*$. Contradiction. Hence, in the worst case, $P_1^{H3} > 0.5$ and $zr' < 0.5$. Therefore, $zr' \le 0.5 - \varepsilon$ from Example 1. Moreover, the bound is tight because of the instance in Example 1. □

When there exist two gaps in σ^d, an optimal solution value $z^* = \max\{(g_1 - P_1^*), (g_1 + g_2 - P_2^*)\}$ where P_i^* is the total processing time of new jobs scheduled at G_i and all earlier gaps in σ^* for $i = 1, 2$. Similarly, $z^{H3} = \max\{(g_1 - P_1^{H3}), (g_1 + g_2 - P_2^{H3})\}$ where P_i^{H3} is the total size of new jobs scheduled at G_i and all earlier gaps in σ^{H3} for $i = 1, 2$. Finally, $z^{H0} = \max\{g_1, (g_1 + g_2)\} = g_1 + g_2$. For convenience, we define the following situation which can occur while solving the problem.

Big Job Situation (BGS). in σ, $E_j(\sigma) \ge \sum_{i=1}^{s} g_i/2$ for some $G_s \in G$ and all $j \in R_s$.

We say that *BGS occurs at G_s in a schedule* or *a schedule has BGS at G_s* if jobs in R_s has earliness cost greater than or equal to $\sum_{i=1}^{s} g_i/2$. The following lemma establishes that if there exist two gaps in σ^d, then the worst case bound on the modified z-approximation for H3 is no worse than the problem with one gap.

Lemma 7. If there exist two gaps in σ^d, then the worst

case bound on the modified z-approximation for H3 is no worse than that for the problem with one gap.

Proof. Observe that if there exists no BGS at G_1 and G_2 in σ^{H3}, then the result holds from the definition of the BGS. Suppose that we are following the steps in H3. From Lemma 6, $r_1P_1^{H3}/(r_1P_1^*) < 0.5$ unless there exists no job to schedule at G_1 in σ^{H3}. If there exists no job to be scheduled at G_1 in σ^{H3}, then neither does it in σ^*. For this case, there must exist a worse case example with the problem with only one gap. Hence, we assume that some jobs are scheduled at G_1 in σ^{H3}.

At this point, if the BGS does not occur at G_2 in σ^{H3}, then it can be seen that the result holds. Hence, we suppose that the BGS occurs at G_2, and thus, the remaining gap at G_2 is greater than or equal to $(g_1+g_2)/2$ in σ^{H3}. Notice that if σ^{H3} has the BGS, then σ^* should have the BGS. Now, if there exists no big job, which is bigger than $(g_1+g_2)/2$, to be scheduled at G_2 in σ^{H3}, then the same situation should happen in σ^*, and thus, $zr' < 0.5$. Hence, we only consider the case where there exists at least one big job, which is bigger than $(g_1+g_2)/2$, to be scheduled at G_2 in σ^{H3}.

First, suppose that an optimal schedule does not schedule any big job at G_2. This implies that $P_2^* \le (g_1+g_2)/2$ and thus, P_2^{H3} should be no less than P_2^* due to existence of the new jobs scheduled in σ^*. Alternatively suppose that big job $r \in N_n$ such that $p_r > (g_1+g_2)/2$ is scheduled at G_2 in σ^*. Note that $P_2^{H3} = p_r > (g_1+g_2)/2$. Since $P_2^* - P_2^{H3} < (g_1+g_2)/2$ and $P_2^* > (g_1+g_2)/2$, $\{(g_1+g_2-P_2^{H3}) - (g_1+g_2-P_2^*)\}/\{(g_1+g_2) - (g_1+g_2-P_2^*)\} < 0.5$.

After H3 schedules job r at G_2 and unschedules new jobs in G_1, the size of the idle time at G_1 is $g_1+g_2-p_r$. Since H3 schedules the smallest big job available at G_2, the size of the idle time at G_1 in σ^{H3} should be no less than that in σ^*. If at least one job is scheduled at G_1 in σ^*, then $P_1^* < P_1^{H3}/2$ from Lemma 6. Alternatively, if no job is scheduled at G_1 in σ^*, then at least one job should be scheduled at G_1 in σ^{H3}, and thus $P_1^* < P_1^{H3}$. Recall that if no job is scheduled at G_1 in σ^{H3}, then there exists a worse example with the problem with only one gap.

For any zr' value generated from the problem with two gaps, we can create a slightly worse case example by reducing ε in Lemma 6. Therefore, the worst case bound on the z-approximation of H3 for problem $1|\ disrupt, no-idle, no-tardy\ |E_{max}$ with the two gaps in the disrupted schedule is no worse than that for the problem with one gap. □

When there exist more than two gaps in the disrupted schedule, an optimal solution value $z^* = \max\{(g_1 - P_1^*), (g_1+g_2-P_2^*), \cdots, \sum_{i=1}^{q} g_i - P_i^*\}$ where P_i^* is the total processing time of new jobs scheduled at G_i and all earlier gaps in σ^* for $i = 1, 2, \cdots, q$. Similarly, $z^{H3} = \max\{(g_1 - P_1^{H3}), (g_1+g_2-P_2^{H3}), \cdots, \sum_{i=1}^{q} g_i - P_i^{H3}\}$ where P_i^{H3} is the total size of new jobs scheduled at G_i and all earlier gaps in σ^{H3} for $i = 1, 2, \cdots, q$. Finally, $z^{H0} = \max\{g_1, (g_1+g_2), \cdots, \sum_{i=1}^{q} g_i\} = \sum_{i=1}^{q} g_i$.

The following lemma establishes that if there exist more than two gaps in σ^d, then the worst case bound on the z-approximation for H3 is no worse than that for the problem with one gap.

Lemma 8. If there exist more than two gaps in σ^d, then the worst case bound on the z-approximation for H3 is no worse than that for the problem with one gap.

Proof. For the problem with more than two gaps in σ^d, suppose that there exists a worst case example where $zr' \ge 0.5$. Then, there exists G_s such that $(\sum_{i=1}^{s} g_i - P_i^{H3}) / \sum_{i=1}^{q} g_i \ge 0.5$.

Notice that earliness cost for jobs in R_s is larger than $\sum_{i=1}^{s} g_i/2$, and it implies that σ^{H3} has the BGS at G_s and there does not exist a big job, which is bigger than $\sum_{i=1}^{s} g_i/2$ and can be scheduled at G_s in σ^{H3}. This also implies that G_s in σ^* also has the same situation. If some additional jobs are scheduled in σ^*, then the same jobs would have been scheduled in σ^{H3}. Hence, zr' should be less than 0.5.

If there exists a big job r such that $p_r > \sum_{i=1}^{s} g_i/2$, then job r would have been scheduled at G_s in σ^{H3}, and it automatically implies that $(\sum_{i=1}^{s} g_i - P_i^{H3})/\sum_{i=1}^{q} g_i < 0.5$. Scheduling a big job may create another BGS at one of the earlier gaps. Then, for each gap $\ell \in \{2, 3, \cdots, s-1\}$, we can use the same argument used for G_s to prove that $(\sum_{i=1}^{\ell} g_i - P_i^{H3})/\sum_{i=1}^{q} g_i < 0.5$. If we eventually create a BGS at G_2, then we can use the argument in the proof of Lemma 7. Therefore, the worst case bound on the modified z-approximation for H3 is no worse than the problem with one gap. □

Theorem 5. For problem $1|\ disrupt, no-idle, no-tardy\ |E_{max}$, $zr' = (z^{H3} - z^*)/(z^{H0} - z^*) \le 0.5 - \varepsilon$ for some $\varepsilon > 0$ and this bound is tight.

***Proof*:** The result follows from Lemmas 6, 7, and 8. □

8. COMPUTATIONAL STUDY

In this section, heuristics H1, H2, and H3 are empirically evaluated. The heuristics are tested on a problem whose objective is to minimize E_{max}. Because finding z^* is computationally intensive, we use a lower bound $z^L = 0$. Hence, the performance indicators for H1, H2, and H3 are the upper bounds on the modified z-approximation, z^{H1}/z^{H0}, z^{H2}/z^{H0}, and z^{H3}/z^{H0}, respectively.

We compare the performances of H1, H2, and H3 under various conditions. We also observe the impact of different factors such as n_o, n_c, n_n, and p_j. For each problem instance, $p_j : DU[p^{LB}, p^{UB}]$ where p^{LB} and p^{UB} are parameters and where $DU[\ell, u]$ is a discrete random variable uniformly distributed between ℓ and u. For a given set of test problems, n_o, n_c, and n_n are fixed. We generate 1,350 test problems under 45 conditions for the problem.

To test the effects of varying the number of original jobs n_o, three different values of n_o are considered: 10,

50, and 100. To determine the effect of varying the number of canceled jobs n_c, three different values of n_c are considered for each of the three n_o values: $0.1n_o$, $0.2n_o$, and $0.3n_o$. To observe the impact from different number of new jobs, we consider three different values of n_n for each of the three n_c values: n_c, $2n_c$, and $3n_c$. It is also possible that the standard deviation of the p_{ij}'s may affect the heuristic performance. Consequently, we consider three different distributions for p_j, $j \in \{1, 2, \cdots, n_o + n_n\}$: $p_j : DU[1, 99]$, $p_j : DU[25, 75]$ and $p_j : DU[40, 60]$. The standard deviations are 28.88, 14.43, and 5.77, respectively.

To avoid excessive testing, $p_j : DU[1, 99]$ and $n_n = 2n_c$ are used as default parameters. For instance, when we test the effects of varying the number of new jobs with $n_o = 100$ and $n_c = 20$, only the instances with $p_j : DU[1, 99]$ and $n_n = 40$ are used. The mean zr' is the arithmetic mean of the modified z-approximation. The mean modified z-approximation is calculated over 30 instances of each problem type. The program is implemented in C language and run on the PC with a Core 2 Duo processor and 2.53 GHz plus 2 GB RAM.

In Table 1, we present the mean modified z-approximation for H1, H2, and H3 where z^L is used as a lower bound for $z^* = 0$. The results in the table are presented side by side for comparison of the heuristics. The results indicate that performances of the heuristics are much better than H0 which represents the schedule where all new jobs are scheduled at the end of a disrupted schedule. The mean modified z-approximation of the all heuristics becomes smaller as n_o increases, and with the same n_o value, it becomes smaller as n_c increases. Hence, we may conclude that the performance of H1, H2, and H3 gets better as n_o increases, and with the same n_o value, the heuristics performs better as n_c increases.

Table 1. Performances of the heuristics

$p_j : DU[1, 99]$			Mean modified z-approximation		
n_o	n_c	n_n	H1	H2	H3
10	1	2	0.6198	0.5831	0.5831
	2	4	0.2711	0.1977	0.1977
	3	6	0.1778	0.1155	0.1155
50	5	10	0.1492	0.0984	0.0973
	10	20	0.0528	0.0253	0.0253
	15	30	0.0339	0.0135	0.0135
100	10	20	0.0571	0.0316	0.0315
	20	40	0.0229	0.0101	0.0101
	30	60	0.0146	0.0055	0.0055

The mean modified z-approximation of H2 and H3 is clearly smaller than that of H1 for each cell where n_o, n_c, and n_n are fixed. Consequently, H2 and H3 perform better than H1. Regarding the comparison between H2 and H3, H3 is only slightly better than H2 for the two cases where $n_o = 50$, $n_c = 5$, and $n_n = 10$ and $n_o = 100$, $n_c = 10$, and $n_n = 20$. For the rest of the cases, their performance seems to be identical. This pattern is consistent throughout the rest of results except for one case with $n_o = 10$, $n_c = 3$, and $n_n = 3$ where H2 performs slightly better than H3 (Table 2).

Table 2. Performances of the heuristics with various n_n when $n_o = 10$

$p_j : DU[1, 99]$			Mean modified z-approximation		
n_o	n_c	n_n	H1	H2	H3
10	1	1	0.5434	0.5434	0.5434
	2	2	0.4692	0.4490	0.4490
	3	3	0.3390	0.2839	0.2990
10	1	2	0.6198	0.5831	0.5831
	2	4	0.2711	0.1977	0.1977
	3	6	0.1778	0.1155	0.1155
10	1	3	0.4645	0.4338	0.4338
	2	6	0.2389	0.1548	0.1453
	3	9	0.1455	0.0869	0.0869

For the same n_o and n_c values, we vary n_n to see the effect of different n_n values. The results are presented in Tables 2–4, and they clearly indicate that for the all heuristics, the mean modified z-approximation becomes smaller as n_n increases. In other words, if there exist more new jobs to schedule at gaps, then the performances of H1, H2, and H3 improve.

Table 3. Performances of the heuristics with various n_n when $n_o = 50$

$p_j : DU[1, 99]$			Mean modified z-approximation		
n_o	n_c	n_n	H1	H2	H3
50	5	5	0.2507	0.2306	0.2298
	10	10	0.1549	0.1367	0.1367
	15	15	0.1317	0.1185	0.1185
50	5	10	0.1492	0.0984	0.0973
	10	20	0.0528	0.0253	0.0253
	15	30	0.0339	0.0135	0.0135
50	5	15	0.0860	0.0539	0.0539
	10	30	0.0334	0.0147	0.0147
	15	45	0.0191	0.0075	0.0075

Table 4. Performances of the heuristics with various n_n when $n_o = 100$

$p_j : DU[1, 99]$			Mean modified z-approximation		
n_o	n_c	n_o	H1	H2	H3
100	10	10	0.1443	0.1345	0.1336
	20	20	0.1023	0.0927	0.0927
	30	30	0.0665	0.0535	0.0535
100	10	20	0.0571	0.0316	0.0315
	20	40	0.0229	0.0101	0.0101
	30	60	0.0146	0.0055	0.0055
100	10	30	0.0399	0.0149	0.0149
	20	60	0.0157	0.0050	0.0050
	30	90	0.0086	0.0021	0.0021

For the same n_o, n_c, and n_n values, the standard deviation of p_j is varied to see the impact of it on the performance of the heuristics. The results in Table 5 indicate that the mean modified z-approximation becomes smaller as the standard deviation of p_j decreases when $n_o = 10$. In other words, the performance of the heuristics gets better as the standard deviation of p_j decreases. However, the same pattern of results cannot be observed in Tables 6 and 7 where $n_o = 50$ and $n_o = 100$, respectively except when $n_o = 50$, $n_c = 5$, and $n_n = 10$.

Table 5. Performances of the heuristics with various standard deviations of processing time when $n_o = 10$

$n_o = 10$			Mean modified z-approximation		
p_j :	n_c	n_n	H1	H2	H3
DU[1, 99]	1	2	0.6198	0.5831	0.5831
	2	4	0.2711	0.1977	0.1977
	3	6	0.1778	0.1155	0.1155
DU[25, 75]	1	2	0.5096	0.4725	0.4725
	2	4	0.2182	0.1537	0.1537
	3	6	0.1390	0.1392	0.1392
DU[40, 60]	1	2	0.4924	0.4043	0.4043
	2	4	0.1934	0.1535	0.1535
	3	6	0.1323	0.1362	0.1362

Table 6. Performances of the heuristics with various standard deviations of processing time when $n_o = 50$

$n_o = 50$			Mean modified z-approximation		
p_j :	n_c	n_n	H1	H2	H3
DU[1, 99]	5	10	0.1492	0.0984	0.0973
	10	20	0.0528	0.0253	0.0253
	15	30	0.0339	0.0135	0.0135
DU[25, 75]	5	10	0.1145	0.0875	0.0875
	10	20	0.0509	0.0387	0.0387
	15	30	0.0355	0.0241	0.0241
DU[40, 60]	5	10	0.1029	0.0805	0.0805
	10	20	0.0488	0.0506	0.0506
	15	30	0.0421	0.0443	0.0443

Table 7. Performances of the heuristics with various standard deviations of processing time when $n_o = 100$

$n_o = 100$			Mean modified z-approximation		
p_j :	n_c	n_n	H1	H2	H3
DU[1, 99]	10	20	0.0571	0.0316	0.0315
	20	40	0.0229	0.0101	0.0101
	30	60	0.0146	0.0055	0.0055
DU[25, 75]	10	20	0.0514	0.0322	0.0318
	20	40	0.0240	0.0146	0.0146
	30	60	0.0176	0.0130	0.0130
DU[40, 60]	10	20	0.0550	0.0366	0.0364
	20	40	0.0320	0.0265	0.0265
	30	60	0.0235	0.0231	0.0231

In other words, the performance of the heuristics does not show a specific pattern except for the cases where $n_o = 50$, $n_c = 5$, and $n_n = 10$. Seeing no particular pattern in general seems to be somewhat surprising because we expected to see the improvement of the performance as the standard deviation of p_j decreases. However, the results imply that the impact of different standard deviations of p_j on the performance is marginal for problems with larger n_o.

9. SUMMARY AND FURTHER RESEARCH

In this paper, we explored a new rescheduling problem where the original objective is minimizing makespan and the disruption related objective is minimizing maximum earliness without tardiness. For this problem, we established the complexity of the problem, and developed three simple but intuitive heuristics. For each of the three heuristics, we found a tight bound on the modified z-approximation ratio. Heuristics H2 and H3 performs much better than H1. Furthermore, H3 is designed to handle some of worst case scenarios better than H2, and it is better than H2 in terms of the worst case bound analysis. However, the performance of H3 is only slightly better than H2 in the computational study.

We believe that major contributions of this paper include the identification of a new rescheduling problem, the proofs of the complexity, and the development and analysis of the heuristics. There are several important possible extensions of this research. The single machine case with the original objective of minimizing makespan was chosen partially due to its tractability of analysis. Hence, it would be worthwhile to consider different original objectives. Also worth considering are different disruption related objective functions and multiple machine environments. Finally, it would be a valuable work to consider the situations with different types of order disruptions such as changes in order priority and due dates.

ACKNOWLEDGMENT

This work was supported by the 2011 sabbatical year research grant of the University of Seoul.

REFERENCES

Azizoglu, M., Koksalan, M., and Koksalan, S. K. (2003), Scheduling to minimize maximum earliness and number of tardy jobs where machine idle time is allowed, *Journal of the Operational Research Society*, **54**(6), 661-664.

Clausen, J., Larsen, J., Larsen, A., and Hansen, J. (2001), Disruption management-operations research between

planning and execution, *OR/MS Today*, **28**(5), 40-43.
Graham, R. L., Lawler, E. L., Lenstra, J. K., and Rinnooy Kan, A. H. G. (1979), Optimization and approximation in deterministic sequencing and scheduling: a survey, *Annals of Discrete Mathematics*, **5**, 287-326.
Garey, M. R. and Johnson, D. S. (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness,* Freeman, San Francisco, CA.
Guner, E., Erol, S., and Tani, K. (1998), One machine scheduling to minimize the maximum earliness with minimum number of tardy jobs, *International Journal of Production Economics*, **55**(2), 213-219.
Hall, N. G. and Potts, C. N. (2004), Rescheduling for new orders, *Operations Research*, **52**(3), 440-453.
Hassin, R. and Khuller, S. (2001), z-Approximations, *Journal of Algorithms*, **41**(2), 429-442.
Johnson, D. S., Demers, A., Ullman, J. D., Garey, M. R., and Graham, R. L. (1974), Worst-case performance bounds for simple one-dimensional packing algorithms, *SIAM Journal on Computing*, **3**(4), 299-325.
Karp, R. M. (1972), Reducibility among combinatorial problems. In: Miller, R. and Thatcher, J. W. (eds.), *Complexity of Computer Computations*, Plenum Press, New York, NY.
Kopanos, G. M., Capon-Garcia, E., Espuna, A., and Puigjaner, L. (2008), Costs for rescheduling actions: a critical issue for reducing the gap between scheduling theory and practice, *Industrial and Engineering Chemistry Industry*, **47**(22), 8785-8795.
Mandel, M. and Mosheiov, G. (2001), Minimizing maximum earliness on parallel identical machines, *Computers and Operations Research*, **28**(4), 317-327.
Molaee, E., Moslehi, G., and Reisi, M. (2010), Minimizing maximum earliness and number of tardy jobs in the single machine scheduling problem, *Computers and Mathematics with Applications*, **60**(11), 2909-2919.
Ozlen, M. and Azizoglu, M. (2011), Rescheduling unrelated parallel machines with total flow time and total disruption cost criteria, *Journal of the Operational Research Society*, **62**, 152-164.
Qi, X., Bard, J. F., and Yu, G. (2006), Disruption management for machine scheduling: the case of SPT schedules, *International Journal of Production Economics*, **103**(1), 166-184.
Unal, A. T., Uzsoy, R., and Kiran, A. S. (1997), Rescheduling on a single machine with part-type dependent setup times and deadlines, *Annals of Operations Research*, **70**, 93-113.
Wu, S. D., Storer, R. H., and Chang, P. C. (1993), One-machine rescheduling heuristics with efficiency and stability as criteria, *Computers and Operations Research*, **20**(1), 1-14.
Yang, J. and Posner, M. E. (2012), *Rescheduling with order disruptions* (Working Paper), The Ohio State University, Columbus, OH.