2.8 Aerial Refueling Process Rescheduling Under Job Related Disruptions

Aerial Refueling Process Rescheduling Under Job Related Disruptions

Sezgin Kaplan & Ghaith Rabadi
Old Dominion University
skaplan@odu.edu grabadi@odu.edu

Abstract. The Aerial Refueling Scheduling Problem (ARSP) can be defined as determining the refueling completion times for each fighter aircraft (job) on the multiple tankers (machines) to minimize the total weighted tardiness. ARSP assumes that the jobs have different release times and due dates. The ARSP is dynamic environment and unexpected events may occur. In this paper, rescheduling in the aerial refueling process with a finite set of jobs will be studied to deal with job related disruptions such as the arrival of new jobs, the departure of an existing job, high deviations in the release times and changes in job priorities. In order to keep the stability and to avoid excessive computation, partial schedule repair algorithm is developed and its preliminary results are presented.

1.0 INTRODUCTION

Aerial refueling (AR) is the process of transferring fuel from one aircraft (the tanker) to another receiver aircraft during flight. The aerial refueling scheduling problem (ARSP) can be modeled as a parallel machine scheduling problem in which we need to determine which jobs have to be allocated to which machines and the sequence of the jobs allocated to each machine. ARSP aims to determine the starting and completion times of refueling process of each receivers on the tankers. It represents a system with *m* identical machines in parallel and *n* jobs where job *j* arrives (becomes available) at ready time r_j and should be complete and leave by the due date d_j.

There are some difficulties that make ARSP different from an ordinary parallel machine scheduling problem. One of these difficulties is sourced from a dynamic environment of the aerial refueling process where disruptions caused by dynamic and unexpected events require rescheduling to update the existing aerial refueling schedule.

In this paper, the parallel machine rescheduling problem with the multiple objectives of minimizing the total weighted tardiness and minimizing schedule instability will be addressed. It is assumed that schedules will be updated only as a result of job related disruptions such as the arrival of new jobs, the departure of an existing job, high deviations in the release times and changes in job priorities. The job related disruptions require a partial rescheduling procedure that aims to change only the affected part of the schedule in order to keep the stability and to avoid excessive computation. As a specific implementation of this procedure, a partial schedule repair algorithm for job arrival disruption has been developed.

The rest of this paper is organized as follows. In Section 2, the related research is summarized. In Section 3, a general rescheduling methodology for job related disruptions is explained. The partial schedule repair algorithm for job arrival disruption is introduced and a sample problem is given in Section 4. Comparison of the partial rescheduling with the regeneration (complete) rescheduling is described in Section 5, and finally results are concluded in Section 6.

2.0 RELATED WORKS

In the literature, rescheduling is required as a result of different disruptions and events such as new (rush) job arrivals, order cancellations, machine failure, changes in order priority, change in ready times, processing time delays, rework due to quality problems, material shortage, operator absenteeism, tool unavailability, due date changes, job order amount changes.

A few researches on the job related disruptions in parallel machine rescheduling can be listed to include Church and Uzsoy [1], Curry and Peters [2], Duenas and Petrovic [3]. Church and Uzsoy [1] introduced a combined periodic and event-driven approach. They developed worst-case error bounds for the periodic approach assuming that an optimal algorithm is used to reschedule the jobs available at each rescheduling point. Ref. [2] considered identical parallel machine scheduling problem with stepwise increasing tardiness cost objectives. Schedule nervousness increases when a scheduling procedure reassigns many planned operations to different machines or different start times. Their measure of schedule nervousness is the proportion of rescheduled jobs that change machine assignment. Ref. [3] presented a new predictive-reactive approach to identical parallel machine scheduling problem with material shortage and job arrival as an uncertain disruption. Their approach is based on generating a predictive schedule using dispatching rules to minimize the makespan. Two rescheduling methods namely left-shifting and building new schedules have been applied. The instability is measured as the starting time deviations between the predictive schedule and the reactive schedule.

Parallel machine rescheduling problems generally have multiple objectives: the objective of the original problem (e.g. minimization total weighted tardiness in our case) and the minimization of the difference between the new schedule (after rescheduling) and the old or initial schedule (before rescheduling).

3.0 RESCHEDULING METHODOLOGY

Three rescheduling approaches continuous, periodic and event-driven were defined by Ref. [1]. Continuous rescheduling takes a rescheduling action each time an event occurs. Periodic rescheduling defines rescheduling points between which any events that occur are ignored until the following rescheduling point. Finally, in the event-driven rescheduling, a rescheduling action is initiated upon an event with potential to cause significant disruption. Both continuous and periodic rescheduling can be viewed as special cases of event-driven rescheduling. In the ARSP, we take continuous and event-driven rescheduling approach where updating the existing schedule should take place when a rare event occurs. Rare events that have a potential to cause significant disruptions in the ARSP are interpreted by the arrival of new jobs, departure of an existing job, high deviations in the release times and changes in job priorities. Processing times and due date tightness are assumed fixed during the scheduling horizon.

There are generally three rescheduling repair methods: right shift rescheduling, partial rescheduling and regeneration. Right shift rescheduling postpones each remaining operation by the amount of time needed to make the schedule feasible. Partial rescheduling algorithm reschedules only the operations affected directly or indirectly by disruption. Regeneration reschedules the entire set of operations not processed before the rescheduling point, including those not affected by the disruption [1]. In the ARSP, a partial schedule repair procedure is developed to keep the unaffected part of the schedule and repair the later affected part by a dispatching rule.

The objective of the aerial refueling rescheduling problem is not only minimizing total weighted tardiness, but also minimizing schedule instability. Tardiness is defined as *max (0, C_j – d_j)* where C_j is completion time, d_j is the due date of job *j* and the total weighted tardiness is defined as $\sum w_jT_j$. Instability of the aerial refueling schedules is defined here as any changes in starting time on the assigned machine for each job. Then the measure of schedule instability can be defined as the proportion of rescheduled jobs that change machine assignment and starting time. Thus, in this paper, a heuristic algorithm that combines an appropriate dispatching rule for the weighted tardiness objective and partial repairing algorithm for schedule stability objective will be introduced.

3.1 General Partial Rescheduling Procedure

If we assume that one of the events occurs at time *t*, a general partial rescheduling procedure can be presented briefly as follows:

Step 1. Update weight and release time vectors (*W* and *R*) according to the event type and initialize processing time and due date input for old and new jobs

Step 2. Determine time *t'* to start repairing

Step 3. Determine *U* for time *t'*

Step 4. Repair the part of the schedule by assigning *U* using an appropriate dispatching rule

Step 5. Calculate objective function values according to the repaired schedule

U is the set of jobs that will be rescheduled point *t'*. *U* selection criterion to include jobs in the set of *U*, should keep stability of the current schedule meanwhile considering the weighted tardiness cost. The part of the existing schedule before *t'* will be kept as is to maintain schedule stability. In the ARSP, we use ATC rule priority index that will be explained in the next section, as a *U* selection criterion.

In Step 2, only weight (*W*) and release time (*R*) vectors are updated because any potential event type can be represented by an update in the weight and release time value of the job in the vector as given Table 1. *M* is a large number to represent presence or absence of a job in the scheduling environment. Due date vector is assumed to be defined as a function of processing time (*P*) and release time (R) vectors with fixed due date tightness, $D = R + \alpha . P$.

Table 1. Weight and release time changes for various event types

j : index of the job causing a disruption	Current Weight (*W*)	Updated Weight (*W'*)	Current Release Time (*R*)	Updated Release Time (*R'*)
Job Arrival	0	w_j	*M*	r_j
Job Departure	w_j	0	r_j	*M*
Changing Release Time	w_j	w_j	r_j	r_j'
Changing Priorities	w_j	w_j'	r_j	r_j

According to Table 1, if job arrival event occurs, *W* can be updated to *W'* by changing the zero weight value to assigned weight value (w_j) for the new job. Moreover, *R* can be updated to *R'* by changing *M* value (i.e. the corresponding job is not considered in the scheduling environment), to release time (r_j) for the new job. Changing release time and changing priority events can be illustrated by updating r_j to r_j' and w_j to w_j' respectively.

3.2 Apparent Tardiness Cost Rule

Dispatching (or Priority) Rules are the most common heuristics for scheduling problems due to their easy implementation and low computational power requirements. Apparent Tardiness Cost (ATC) heuristic is a good composite dispatching rule for the parallel machine total weighted tardiness problem. It combines the WSPT (Weighted Shortest Processing Time) rule and the Minimum Slack First rule (the job with the minimum slack is scheduled first) [4]. The priority index of ATC is defined as

$$\pi_j(t) = \frac{w_j}{p_j} \exp\left[- \frac{\max[d_j - p_j - t, 0]}{k\overline{p}} \right] \qquad (1)$$

where

w_j : weight of the remaining job j

p_j : processing time of the remaining job j

d_j : due date of the remaining job j

t: Decision time point that the resource is considering which job to choose next.

$\bar{p}$: The average processing time of the remaining jobs,

k: 'look-ahead' or planning parameter and is set empirically (*k*= 2 is used here).

$S_j^+(t)$: the slack factor is equal to $max\,[d_j - p_j - t, 0]$.

This priority index is a function of the time t at which the machine become free. Under the ATC rule jobs are scheduled one at a time; that is, every time the machine becomes free the ranking index is computed for each remaining job. The job with the highest ranking index is then selected to be processed next.

4.0 PARTIAL RESCHEDULING ALGORITHM FOR JOB ARRIVAL DISRUPTION

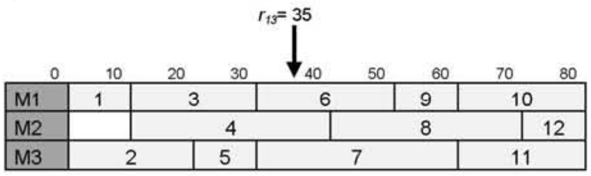
A partial rescheduling algorithm using the general procedure mentioned in Section 3 is developed for job arrival disruption and pseudo code of the algorithm is given as follows:

r_{new}: arrival time of the new job
r_j: release time of the job *j*
p_j: processing time of the job *j*
w_j weight of the job *j*
d_j: due date of the job *j*
$StartTime_j$: start time of the job *j*
$CompTime_j$: completion time of the job *j*
$Cmax_i$: makespan of the machine *i*
$\pi_j\,(t)$: priority of the job *j* at decision time *t*
S: set of jobs in the old schedule
U: set of waiting jobs to be rescheduled
C: set of jobs whose $\pi_j(t) > \pi_{new}(t)$
M: set of machines

```
1. Initialize $r_j$, $p_j$, $w_j$, $d_j$ $\forall j \in S$ and for the new job
2. Set $StartTime_j$ and $CompTime_j$ $\forall j \in S$ and
   $StartTime_{new}$ = large integer, $CompTime_{new}$ = large integer for the new job
3. Determine $Cmax_i$ = first completion time on machine $i$ after rnew, $\forall i \in M$
4. Set $t = \min_{i \in M} \{ Cmax_i \}$
5. Set $U=\{U \subseteq (S \cup \text{new job}) : StartTime_j \geq t \;\; \forall j \in U \}$
6. Calculate $\pi j(t)$ $\forall j \in U$ by using ATC
7. Determine set $C = \{ j \in C : \pi j(t) > \pi_{new}(t) \}$
8. Update $Cmax_i$ according to the latest $CompTime_j$ on machine $i$, $\forall i \in M$ and $j \in C$
9. if $StartTimej < Cmax_i$ $\forall j \in S$ and $\forall i \in M$ then remove $j$ from $U$
10. Update $t = \min_{i \in M} \{ Cmax_i \}$ ,
11. while $U \neq \emptyset$
12.     Find $j=\{j \in U : \pi j(t) = \max_{k \in U} \{\pi k\} \}$
13.     Find $i = \{i \in M : Cmax_i\,(t)= \min_{m \in M}\{ Cmax_m \} \}$
14.     Update $CompTime_j = Cmax_i + max(r_j, t) + p_j$ and remove $j$ from $U$
15.     Update $Cmax_i = CompTime_j$
16.     Update $t = \min_{i \in M} \{ Cmax_i \}$
17.     Calculate $\pi j(t)$ $\forall j \in U$ by using ATC
18. end while
19. Calculate and display the objective values, TWT and Instability.
```

The set *U* and start of the schedule repair are determined by steps 3-10. ATC rule priority index is used to determine the jobs in set *U* as a selection. The main rationale to determine set *U* is that jobs in set *C* (set of jobs whose $\pi_j(t) > \pi_{new}(t)$) cannot be scheduled later than new arriving job. Processes are assumed non-preemptive while determining decision time, *t*. Steps 11-18 is the repairing part of the algorithm that assigns the jobs to the machines by using ATC. In order to explain the partial rescheduling algorithm, a sample problem is given as follows :

A new job (the 13th job for the instance with number of jobs = 12, number of machines = 3) at time t = 35.



Step 1. Input data for an instance.

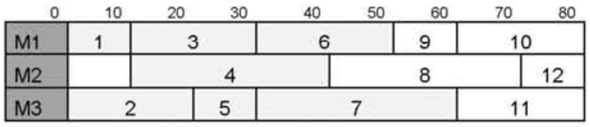
j	1	2	3	4	5	6	7	8	9	10	11	12	13
rj	0	0	10	10	20	30	30	40	50	60	60	70	35
pj	10	20	20	30	10	20	30	30	10	20	20	10	20
wj	8	7	5	4	8	5	3	9	8	9	6	1	5
dj	20	40	50	70	40	70	90	90	70	100	100	90	75

Step 2. Input start and completion times for set S and large integers for the new job.

job	1	2	3	4	5	6	7	8	9	10	11	12	13
StartTimej	0	0	10	10	20	30	30	40	50	60	60	70	M*
CompTimej	10	20	30	40	30	50	60	70	60	80	80	80	M*

M* = 99999

Step 3,4. Determine Cmaxi for i = {1,2,3} and t.



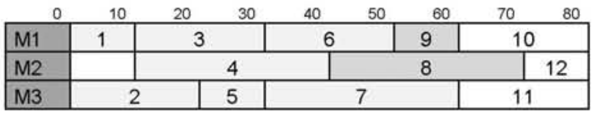
$Cmax_1$ = 50, $Cmax_2$ = 40, $Cmax_3$ = 60, t = min {50, 40, 60} = 40

Step 5. U = {8,9,10,11,12,13}

Step 6. $\pi_9(t)$= 0.609 > $\pi_8(t)$= 0.262 > $\pi_{13}(t)$=0.166 > $\pi_{11}(t)$= 0.152 > $\pi_{10}(t)$= 0.151 > $\pi_{12}(t)$= 0.051

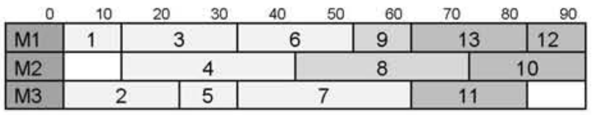
Step 7. C = {8,9} (jobs that have to be scheduled before the new job).

Step 8,9,10. $CompTime_8$ = 60 and $CompTime_9$ = 70



$Cmax_1$ = 60, $Cmax_2$ = 70, $Cmax_3$ = 60, U = {10,11,12,13}, t = min {60, 70, 60} = 60

Step 11-18.



Step 19. TWT = 70, $Instability$ = 3 (Jobs 10, 11, 12)

5.0 COMPUTATIONAL STUDY

To measure the effectiveness of the partial rescheduling algorithm, it is compared with the complete (regeneration) rescheduling algorithm in which all jobs after the arrival the new job are rescheduled using ATC rule similar to the way described in Steps 11-18 of the partial rescheduling algorithm. For this aim, the TWT and Instability (number of jobs whose start time or assigned machine has changed) objective values were obtained for different combinations of release times (10, 30, 50), processing times (15, 30, 45) and weights (1, 5, 9) of the arriving job. Experimental results are given in Table 2. Due dates are generated by equation $d_j = r_j + \alpha.p_j$ with α = 2.

If we assume that instance samples were drawn independently and randomly from the instance population for two algorithms, the single factor ANOVA for comparing the algorithms in TWT shows that the partial rescheduling is superior to the complete rescheduling in terms of TWT at 80% significance level. ANOVA was also performed on the Instability and the partial rescheduling is significantly superior to the complete algorithm.

Table 2. TWT and Instability Values for Partial and Complete Rescheduling

Instance	Release Time	Processing Time	Weight	Total Weighted Tardiness		Instability*	
				Partial Rescheduling	Complete Rescheduling	Partial Rescheduling	Complete Rescheduling
1	10	15	1	261.6	301.7	3	10
2	10	15	5	355.2	371.6	6	10
3	10	15	9	237.7	237.7	10	10
4	10	30	1	208.6	309.5	2	10
5	10	30	5	426.5	461.1	4	10
6	10	30	9	418.9	411.0	6	10
7	10	45	1	193.6	294.5	0	10
8	10	45	5	359.2	470.1	3	10
9	10	45	9	404.5	502.5	4	10
10	30	15	1	190.0	225.5	2	8
11	30	15	5	255.2	232.5	6	8
12	30	15	9	233.5	233.5	8	8
13	30	30	1	188.6	215.0	0	8
14	30	30	5	326.5	272.6	4	8
15	30	30	9	276.6	310.9	6	8
16	30	45	1	173.6	200.0	0	8
17	30	45	5	248.8	301.6	2	8
18	30	45	9	349.7	362.6	4	8
19	50	15	1	170.0	199.0	2	6
20	50	15	5	201.2	201.2	6	6
21	50	15	9	201.2	201.2	6	6
22	50	30	1	168.6	189.7	0	6
23	50	30	5	222.2	292.6	3	6
24	50	30	9	321.1	279.2	4	6
25	50	45	1	153.6	174.7	0	6
26	50	45	5	154.8	191.1	2	6
27	50	45	9	237.2	369.2	3	6

* Instability : Number of jobs whose start time or assigned machine has changed.

6.0 CONCLUSIONS

This paper presents a partial rescheduling algorithm for job arrival disruptions in the parallel machine scheduling. A heuristic algorithm that combines an appropriate dispatching rule (the ATC) for the total weighted tardiness objective with partial repair algorithm for the schedule stability objective, was introduced. This algorithm can be modified to use for other types of job related disruptions. To measure the effectiveness of the partial rescheduling algorithm, it was compared with the complete (regeneration) rescheduling algorithm. According to ANOVA, the partial rescheduling has superior results compared to the complete rescheduling. In future research, more constraints such as machine compatibility, sequence dependent setup times and deadlines may be included in the model. In addition, insertion algorithms that emphasize more the instability objective may be developed. Other composite dispatching rules and metaheuristics may be implemented to obtain better results and linear models may be programmed to find optimal solutions that can be used for comparisons.

7.0 REFERENCES

[1] Church, L.K., and Uzsoy, R., Analysis of periodic and event driven rescheduling policies in dynamic shops, International Journal of Computer Integrated Manufacturing, 5, 153–163, 1992.

[2] Curry, J. and Peters, B., Rescheduling parallel machines with stepwise increasing tardiness and machine assignment stability objectives, International Journal of Production Research,43:15, 3231 – 3246, 2005.

[3] Duenas, A., and Petrovic, D., An approach to predictive-reactive scheduling of parallel machines subject to disruptions, Annual Operations Research, 159, 65–82, 2008.

[4] Pinedo, M., Scheduling theory, algorithms, and systems. 3rd Ed., Springer, New York, 2008.

MODSIM WORLD Conference & Expo

October 13–15, 2010
Hampton, Virginia

Aerial Refueling Process Rescheduling Under Job Related Disruptions

Sezgin KAPLAN, Ph.D. Student
Old Dominion University

October 13, 2010

MODSIM WORLD Conference & Expo

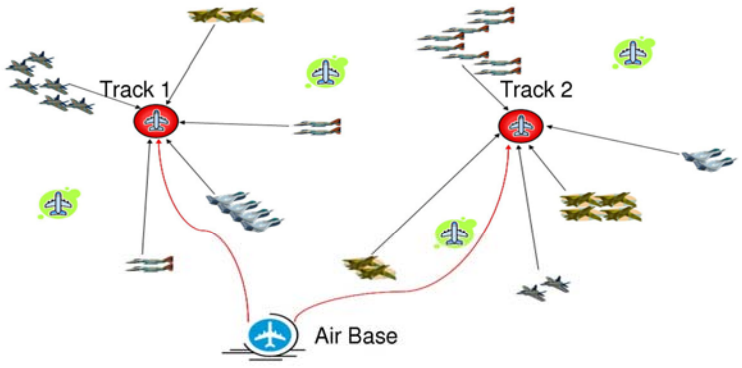
Outline

- Aerial Refueling Scheduling Problem (ARSP)
- Rescheduling Problem
- AR Rescheduling Problem
- Partial Rescheduling Algorithm
- Apparent Tardiness Cost (ATC) Rule
- An Example
- Computational Study
- Conclusion
- References

ARSP

- Aerial Refueling: The process of transferring fuel from one aircraft (a tanker) to another (a fighter as a receiver) during flight.

ARSP



- ARSP: Determining the refueling completion times for each fighter aircraft (job) on one of multiple tankers (machines).

ARSP

- **Model :** An identical parallel machine scheduling problem with release times and due dates.
- A system with *m* identical machines in parallel and *n* jobs where job *j* becomes available at ready time r_j and should be complete and leave by the due date d_j.
- **Objective :** To minimize total weighted tardiness (TWT) as a performance measure.

Rescheduling Problem

- Disruptions caused by dynamic and unexpected events require rescheduling to update the existing aerial refueling schedule.
- **Types of events :** New (rush) job arrivals, order cancellations, machine failure, changes in order priority/ready times/processing times/due dates, rework due to quality problems, material/operator/tool unavailability etc [1,2,3,4].

Rescheduling Problem

- Continuous, periodic and event-driven rescheduling approaches [1].
- Right shift rescheduling, partial rescheduling and regeneration methods.
- **Multi objectives :** objective of the original problem + minimization of the difference between old and new schedules.

AR Rescheduling Problem

- Job related disruptions :
  – the arrival of new jobs,
  – the departure of an existing job,
  – high deviations in the release times,
  – changes in job priorities.
- Continuous and event-driven rescheduling approach.
- Partial schedule repair method.

AR Rescheduling Problem

- **Objective :** minimize TWT + minimize instability.
- **Tardiness :** *max(0,C_j-d_j)* where C_j is completion time, d_j is the due date of job *j.*
- Total weighted tardiness (TWT) : $\sum w_j T_{j.}$
- **Instability :** any changes in starting time on the assigned machine for each job.
- **Measure of instability :** the proportion of rescheduled jobs that change machine assignment and starting time.

Partial Rescheduling Algorithm

- General partial rescheduling procedure can be used for different types of job related disruptions.
  1. Initialize data (ready time, processing time, weight, due date) according to the event type occurred at time *t.*
  2. Determine the part of the schedule to repair.
  3. Repair the part by using an appropriate dispatching rule
- A heuristic algorithm for arrival of new jobs :

  Apparent Tardiness Cost (ATC) dispatching rule + Partial repair algorithm.

Rescheduling Procedure

j : index of the job causing a disruption	Current Weight (W)	Updated Weight (W')	Current Release Time (R)	Updated Release Time (R')
Job Arrival	0	w_j	M	r_j
Job Departure	w_j	0	r_j	M
Changing Release Time	w_j	w_j	r_j	r_j'
Changing Priorities	w_j	w_j'	r_j	r_j

Table 1. Weight and release time changes for various event types

M : a large number to represent presence or absence of a job in the scheduling environment.

ATC Rule

- A good composite dispatching rule for the total weighted tardiness problem [4].
- A dynamic algorithm that after each job completion, the remaining jobs are analyzed, priorities derived according to improved formula, and the highest priority job selected.
- The priority index :

$$\pi_j(t) = \frac{w_j}{p_j}\exp\left[-\frac{\max\left[d_j - p_j - t, 0\right]}{k\overline{p}}\right]$$

  - Urgent job :
    If $C_j \geq d_j$, then $\pi_j = w_j/p_j$

w_j : weight of the remaining job j
p_j : processing time of the job j
d_j : due date of the job j
t : Decision time point
$\overline{p}$: The average processing time of the remaining jobs,
k : 'look-ahead' parameter.
$S_j^{+}(t)$, the slack factor : $max\ [d_j - p_j - t, 0]$.

Partial Rescheduling Algorithm

r_{new}: arrival time of the new job
r_j: release time of the job j
p_j: processing time of the job j
w_j weight of the job j
d_j: due date of the job j
$StartTime_j$: start time of the job j
$CompTime_j$: completion time of the job j
$Cmax_i$: makespan of the machine i
$\pi_j(t)$: priority of the job j at decision time t
S: set of jobs in the old schedule
U: set of waiting jobs to be rescheduled
C: set of jobs whose $\pi_j(t) > \pi_{new}(t)$
M: set of machines

```
1.  Initialize $r_j$, $p_j$, $w_j$, $d_j$ $\forall j \in S$ and for the new job
2.  Set $StartTime_j$ and $CompTime_j$ $\forall j \in S$ and
    $StartTime_{new}$ = large integer, $CompTime_{new}$ = large integer for the new job
3.  Determine $Cmax_i$ = first completion time on machine $i$ after rnew, $\forall i \in M$
4.  Set $t = \min_{i \in M} \{ Cmax_i \}$
5.  Set $U = \{U \subseteq (S \cup \text{new job}) : StartTime_j \geq t \ \forall j \in U \}$
6.  Calculate $\pi j(t)$ $\forall j \in U$ by using ATC
7.  Determine set $C = \{ j \in C : \pi j(t) > \pi_{new}(t) \}$
8.  Update $Cmax_i$ according to the latest $CompTime_j$ on machine $i$, $\forall i \in M$ and $j \in C$
9.  if $StartTimej < Cmax_i$ $\forall j \in S$ and $\forall i \in M$ then remove $j$ from $U$
10. Update $t = \min_{i \in M} \{ Cmax_i \}$ .
11. while $U \neq \emptyset$
12.     Find j=$\{j \in U : \pi j(t) = \max_{k \in U} \{\pi k\} \}$
13.     Find $i = \{i \in M : Cmax_i (t) = \min_{m \in M} \{ Cmax_m \} \}$
14.     Update $CompTime_j = Cmax_i + max(r_j, t) + p_j$ and remove $j$ from $U$
15.     Update $Cmax_i = CompTime_j$
16.     Update $t = \min_{i \in M} \{ Cmax_i \}$
17.     Calculate $\pi j(t)$ $\forall j \in U$ by using ATC
18. end while
19. Calculate and display the objective values, TWT and Instability.
```

Input : Steps 1 and 2

Determine the *U* set to reschedule: Steps 3-10

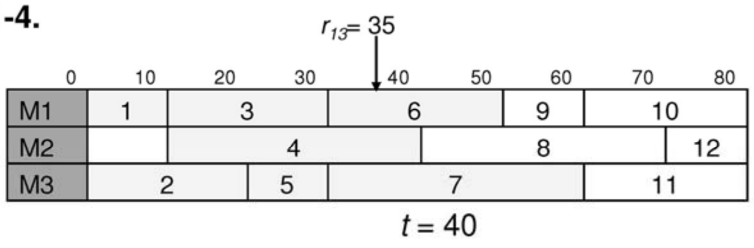
The main rationale : jobs in set *C* (set of jobs whose $\pi_j(t) > \pi_{new}(t)$) cannot be scheduled later than new arriving job.

Reschedule by ATC : Steps 11-18

Display

An Example

Step 1-4.

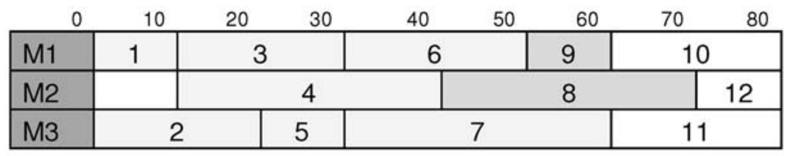


Step 5. U = {8,9,10,11,12,13}

Step 6. $\boldsymbol{\pi_9(40)}$**= 0.609 >** $\boldsymbol{\pi_8(40)}$**= 0.262** > $\pi_{13}(40)$=0.166 > $\pi_{11}(40)$= 0.152 > $\pi_{10}(40)$= 0.151 > $\pi_{12}(40)$= 0.051

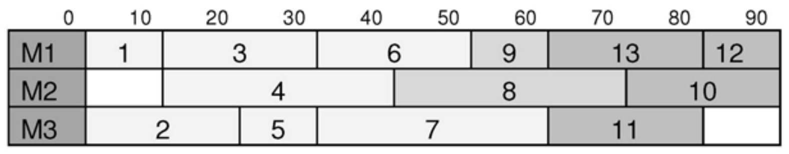
Step 7. C = {8,9} (jobs that have to be scheduled before the new job).

An Example



Step 8-10. $CompTime_8 = 60$ and $CompTime_9 = 70$
$Cmax_1 = 60$, $Cmax_2 = 70$, $Cmax_3 = 60$,
$U = \{10,11,12,13\}$, $t = min\ \{60, 70, 60\} = 60$

Step 11-18. $\pi_{13}(60) > \pi_{11}(60) > \pi_{10}(60) > \pi_{12}(60)$



Step 19. $TWT = 70$, $Instability = 2$ (Jobs 10 and 12)

Computational Study

Table 2. TWT and Instability Values for Partial and Complete Rescheduling

Instance	Release Time	Processing Time	Weight	Total Weighted Tardiness		Instability*	
				Partial Rescheduling	Complete Rescheduling	Partial Rescheduling	Complete Rescheduling
1	10	15	1	261.6	301.7	3	10
2	10	15	5	355.2	371.6	6	10
3	10	15	9	237.7	237.7	10	10
4	10	30	1	208.6	309.5	2	10
5	10	30	5	426.5	461.1	4	10
6	10	30	9	418.9	411.0	6	10
7	10	45	1	193.6	294.5	0	10
8	10	45	5	359.2	470.1	3	10
9	10	45	9	404.5	502.5	4	10
10	30	15	1	190.0	225.5	2	8
11	30	15	5	255.2	232.5	6	8
12	30	15	9	233.5	233.5	8	8
13	30	30	1	188.6	215.0	0	8
14	30	30	5	326.5	272.6	4	8
15	30	30	9	276.6	310.9	6	8
16	30	45	1	173.6	200.0	0	8
17	30	45	5	248.8	301.6	2	8
18	30	45	9	349.7	362.6	4	8
19	50	15	1	170.0	199.0	2	6
20	50	15	5	201.2	201.2	6	6
21	50	15	9	201.2	201.2	6	6
22	50	30	1	168.6	189.7	0	6
23	50	30	5	222.2	292.6	3	6
24	50	30	9	321.1	279.2	4	6
25	50	45	1	153.6	174.7	0	6
26	50	45	5	154.8	191.1	2	6
27	50	45	9	237.2	369.2	3	6

* Instability : Number of jobs whose start time or assigned machine has changed.

Complete (regeneration) rescheduling algorithm : All jobs after the arrival of the new job are rescheduled using ATC rule.

Conclusion

- A heuristic algorithm that combines an appropriate dispatching rule (the ATC) for the total weighted tardiness objective with partial repair algorithm for the schedule stability objective, was introduced.
- To measure the effectiveness of the partial rescheduling algorithm, it was compared with the complete (regeneration) rescheduling algorithm. The partial rescheduling has superior results compared to the complete rescheduling.
- In future research, more constraints such as machine compatibility, sequence dependent setup times and deadlines may be included in the model. In addition, insertion algorithms that emphasize more the instability objective may be developed. Other composite dispatching rules and metaheuristics may be implemented to obtain better results.

References

[1] Church, L.K., and Uzsoy, R., Analysis of periodic and event driven rescheduling policies in dynamic shops, International Journal of Computer Integrated Manufacturing, 5, 153–163, 1992.

[2] Curry, J. and Peters, B., Rescheduling parallel machines with stepwise increasing tardiness and machine assignment stability objectives, International Journal of Production Research,43:15, 3231 – 3246, 2005.

[3] Duenas, A., and Petrovic, D., An approach to predictive-reactive scheduling of parallel machines subject to disruptions, Annual Operations Research, 159, 65–82, 2008.

[4] Pinedo, M., Scheduling theory, algorithms, and systems. 3rd Ed., Springer, New York, 2008.

Questions/Comments