Rolling Partial Rescheduling with Dual Objectives for Single Machine Subject to Disruptions[1)]

WANG Bing[1,2] XI Yu-Geng[2]

[1](*School of Information Engineering, Shandong University at Weihai, Weihai* 264209)
[2](*Institute of Automation, Shanghai Jiaotong University, Shanghai* 200030)
(E-mail: wangbing@sdu.edu.cn)

Abstract This paper discusses the single-machine rescheduling problem with efficiency and stability as criteria, where more than one disruption arises in large-scale dynamic circumstances. Partial rescheduling (PR) strategy is adopted after each disruption and a rolling mechanism is driven by events in response to disruptions. Two kinds of objective functions are designed respectively for PR sub-problem involving in the interim and the terminal of unfinished jobs. The analytical result demonstrates that each local objective is consistent with the global one. Extensive computational experiment was performed and the computational results show that the rolling PR strategy with dual objectives can greatly improve schedule stability with little sacrifice in efficiency and provide a reasonable trade-off between solution quality and computational efforts.

Key words Disruptions, efficiency and stability, partial rescheduling, rolling mechanism

1 Introduction

A vast majority of literature addresses the deterministic scheduling problems, which determine a schedule over a certain time frame assuming all problem characteristics are known. Such schedules are often produced in advance in order to direct production operations and to support other planning activities such as tooling, raw material delivery, and resource allocation. However, unforeseen disruptions such as rush orders, excess processing time, machine breakdown, *etc.* will arise during the execution of such an initial schedule. Such uncertainty is usually dealt with in a reactive rescheduling manner[1].

Two types of reactive rescheduling strategies are normally used in the existing literature. Full rescheduling (FR) strategy, where all unfinished jobs are rescheduled with respect to certain criterion, can be merely applied to problems with moderate size due to computational complexity though it can result in an optimal solution. Right-shift rescheduling (RSR) strategy, where all unfinished jobs are just slid to the right as far as necessary to accommodate the disruption, can not guarantee the solution quality though it requires the least computational efforts. Compromising FR and RSR, partial rescheduling (PR) strategy involve partial unfinished jobs to reschedule. Sabuncuoglu *et al.*[1] adopted PR to deal with large rescheduling problems, where only the schedule performances were considered. However, a practical solution of rescheduling problem requires satisfaction of two often conflicting goals: 1) to retain schedule efficiency, *i.e.* to keep the schedule performance with little deterioration as possible as one can, and 2) to minimize the cost impact of the schedule change, which is referred to as schedule stability. Wu *et al.*[2] addressed such bi-criterion problems with one disruption by use of FR strategy.

During the execution of a large initial schedule, more than one disruption possibly may arise because of the long execution duration, and moreover the number of unfinished operations is likely very large at each disruption. In such situations, FR is neither beneficial nor needed because many unfinished operations may be probably rescheduled more than one time. PR will be a viable alternative. In order to deal with large scheduling problems, Wang *et al.*[3,4] used a rolling horizon mechanism in the deterministic scheduling environments. In this paper, we will use this rolling mechanism to deal with large rescheduling problems with schedule efficiency and schedule stability for single machine subject to more disruptions. During the procedure, the dual objectives are partly considered and realized separately in each PR sub-problem. The consistency between PR local objectives and the global objective will be theoretically discussed. Computational experiment is conducted to testify the theoretical conclusion and the solution qualities.

2 Single-machine rescheduling with efficiency and stability

Consider a single-machine problem with release times to minimize the makespan. There are n jobs to be scheduled. A job denoted as i has a release time r_i, a processing time p_i, and a tail q_i.

1) Supported by National Natural Science Foundation of China (60274013, 60474002) and Science Research Foundation of Shandong University at Weihai (XZ2005001)
Received August 10, 2005; in revised form January 8, 2006

These three parameters of each job are known a priori. For a schedule S of this problem, the makespan denoted as $M(S)$ is defined as follows:

$$M(S) = \max_{i \in S}(b_i + p_i + q_i) \tag{1}$$

where b_i is the beginning time of job i in S. This problem is NP-hard[5].

A minimal makespan initial schedule S^0 can be generated without considering any disruptions. However, after a disruption occurs, at the moment u when the machine returns to service the unfinished jobs, the unfinished jobs should be rescheduled. No matter the disruption is a rush order or an excess processing time, the disruption duration can be conceived as a failure interval of the machine. Therefore, the release times of all unfinished jobs are updated to be r_i' as follows:

$$r_i' = \max(u, r_i) \tag{2}$$

In order to deal with the rescheduling problem with efficiency and stability, we should present the measure for schedule stability. In this paper context, the schedule stability is measured by the schedule deviation of the new schedule S to the initial schedule S^0 just like [2], which is denoted as $D(S)$, *i.e.*

$$D(S) = \sum_{i \in S} |b_i - b_i^0| \tag{3}$$

where b_i^0 is the beginning time of job i in the initial schedule S^0.

The rescheduling problems with dual objectives are to minimize both (1) and (3), which can be converted into a single overall objective by giving two weights to the dual objectives. Let the rescheduling criterion be

$$\min_S \quad J(S) = w_D D(S) + w_M M(S) \tag{4}$$

where w_D is the weight for schedule deviation and w_M is the weight for schedule efficiency. The amount of weight represents the importance degree of the corresponding objective. If a pair of weights is specified, for the optimal solution S^* of the overall objective problem, $(D(S^*), M(S^*))$ is a pair of effective solution of the corresponding bi-criterion problem. If the weights vary, more effective solutions can be obtained. This single objective problem is also NP-hard[2].

We might as well give the same weights to dual objectives. Let the global overall objective be

$$\min_S \quad J(S) = D(S) + M(S) \tag{5}$$

3 PR strategy with dual objectives

In this paper, we explore a dynamic production environment where more disruptions probably occur. We use t to identify a decision point when a disruption occurs and the corresponding rescheduling needs to be performed. Let N be the set of all jobs. The initial schedule S^0 can be generated based on N by simply minimizing the makespan. S^0 is implemented until the first disruption occurs. The first new schedule is obtained through the first rescheduling and is implemented until the next disruption, and so on. Generally, at t, the new schedule obtained through the current rescheduling, denoted as $S(t)$, is referred to as the original schedule of the next rescheduling. Obviously, the original schedule $S(1)$ of the first rescheduling is exactly the initial schedule S^0.

At t, let $\hat{N}_t$ be the set of all finished jobs, which constitute the implemented partial schedule $\hat{S}(t)$. Let u_t be the moment when the system returns to service and D_t be the disruption duration time. If let N_t' be the set of all unfinished jobs, which constitute the unimplemented original schedule $S(N_t')$, then $N = \hat{N}_t \cup N_t'$ and the global original schedule $S(t-1)$ consists of $\hat{S}(t)$ and $S(N_t')$.

When PR strategy is performed at t, a certain number of unfinished jobs from the beginning of $S(N_t')$ are involved in a PR sub-problem, where all jobs are totally rescheduled with respect to a certain criteria.

Definition 1. At t, the set of unfinished jobs involved in the PR sub-problem is referred to as PR horizon, denoted as N_t. The size of PR horizon refers to the number of jobs in N_t, denoted as $|N_t|$.

After a disruption occurs, we do not known how much degree the disruption can interfere the original schedule. The PR horizon can be conceived as a specified transient period to accommodate the disruption, where the job sequence may be changed and idle time may be absorbed. For the remaining

original schedule following the PR horizon, we keep the job sequence unvaried simply by use of RSR strategy. Such rescheduling scheme consisting of two sections can actually be thought that a match-up point is forced on the end of the PR horizon.

Match-up rescheduling (MUR)[6] was proposed by Bean and Birge. When a disruption occurs, rescheduling can make the new schedule completely consistent with the initial schedule from certain point on under certain conditions, *i.e.* the disruption can be accommodated during a transient period. The rescheduling with respect to such a goal is referred to as match-up rescheduling. The point in the initial schedule is referred to as match-up point and the transient period is referred to as match-up time. MUR may request to minimize the match-up time. However, given a match-up point forced on the initial schedule, a new schedule can match up the initial schedule without delay from the match-up point on only if enough idle time exists in the match-up time, and if not, there must be a delay of the actual completion time of match-up point in the new schedule, which is referred to as match-up delay. MUR may request to minimize the match-up delay as well.

The beginning time of a partial schedule refers to the beginning time of the first job in the partial schedule and the completion time of a partial schedule refers to the completion time of the last job in the partial schedule. At t, the partial original schedule for N_t is denoted as $S(N_t)$, whose completion time is denoted as $C(t-1)$. Assume that the new schedule for N_t obtained through PR is denoted as $S^P(N_t)$, whose completion time is denoted as $C^P(t)$. If $S^P(N_t)$ cannot match-up without delay $S(N_t)$ in PR horizon, the match-up delay is $\Delta C^P(t) = C^P(t) - C(t-1)$. Let $\tilde{N}_t$ be the set of the remaining jobs in N_t' excluding N_t, *i.e.* $N_t' = N_t \cup \tilde{N}_t$. The number of jobs in $\tilde{N}_t$ is denoted as $|\tilde{N}_t|$. Two kinds of objective function for PR sub-problem are defined as follows based on N_t respectively locating in the interim or the terminal of original schedule:

$$\min_{S(N_t)} J_t = \{\sum_{i\in B_t} |b_i(t) - b_i(t-1)| + |\tilde{N}|[C(t) - C(t-1)]\}, \quad |\tilde{N}| > 0 \tag{6}$$

$$\min_{S(N_t)} J_t = \{\sum_{i\in N_t} |b_i(t) - b_i(t-1)| + M(S(t))\}, \quad |\tilde{N}| = 0 \tag{7}$$

where $b_i(t)$ and $b_i(t-1)$ represent the beginning times of job i respectively in $S(t)$ and $S(t-1)$. (6) is designed for N_t locating in the interim of original schedule. The objective of PR is to minimize both the match-up delay and the schedule deviation. It is reasonable to use the number of later jobs as the weight for match-up delay in case more idle time greatly puts off later jobs. In such a manner, the schedule deviation of the latter job set $\tilde{N}_t$ is actually considered in the PR sub-problem. When N_t locates in the terminal of original schedule, $\tilde{N}_t$ is empty and we directly consider the makespan in (7). In this paper, the PR objective, just like (6) or (7), is referred to as a local objective, and the objective for global rescheduling, just like (5), is referred to as the global objective. It is shown that the global objective is reflected to some extent in each local objective.

Definition 2. Let the initial schedule of a single-machine scheduling problem be S^0, a Δt-RSR solution of S^0, denoted as S^R, refers to a schedule obtained through RSR, where the beginning time of the first unfinished job is shifted to the right by Δt from that in S^0.

If $S^P(N_t)$ is delayed by $\Delta C^P(t)$ to the original schedule in the PR horizon, the original schedule for $\tilde{N}_t$, denoted as $S(\tilde{N}_t)$, will be shifted to the right by $\Delta C^P(t)$ in order to keep the schedule feasible, *i.e.* the new schedule for $\tilde{N}_t$, denoted as $S^{PR}(\tilde{N}_t)$, is the $\Delta C^P(t)$-RSR solution of $S(\tilde{N}_t)$. When $\Delta C^P(t) = 0$, $S(\tilde{N}_t)$ will be actually kept unvaried. At t, the global rescheduling consists of the PR for N_t and the RSR for $\tilde{N}_t$. The new schedule for N_t', denoted as $S^P(N_t')$, consists of $S^P(N_t)$ and $S^{PR}(\tilde{N}_t)$, and the global new schedule $S(t)$ consists of $\hat{S}(t)$ and $S^P(N_t')$.

Definition 3. This kind of PR considering dual objectives is referred to as partial rescheduling with dual objectives, which is termed for short PRDO.

4 Rolling PRDO as well as analysis of global objective

When more disruptions occur during the execution of an initial schedule, PRDO is driven by disruptions in a rolling mechanism. Let l be the number of disruptions, the rolling PRDO is performed as follows:

Step 1. Minimize the makespan of the problem to generate the original schedule without considering any disruption, and let $S(0) = S^0$, $t = 1$.

Step 2. Implement the original schedule $S(t-1)$ until a disruption occurs, when the time is noted as d_t.

Step 3. For a specified disruption duration D_t, compute the time u_t for the machine returning to service, $u_t = d_t + D_t$, the release times of unfinished jobs in N_t' are updated according to (2) (after the disruption, the interrupted job is resumed and included into N_t').

Step 4. The first k_t jobs from the beginning of $S(N_t')$ are included into the PR-horizon N_t, note the completion time $C(t-1)$, compute the number of jobs in $\tilde{N}_t$, $|\tilde{N}_t| = n - (|\hat{N}_t| + |N_t|)$ (Here k_t is the specified size of PR-horizon).

Step 5. If $|\tilde{N}_t| > 0$, the PR sub-problem with respect to (6) is solved. The solution $S^P(N_t)$, the completion time $C^P(t)$, and the delay $\Delta C^P(t) = C^P(t) - C(t-1)$ can be obtained. The new schedule $S^{PR}(\tilde{N}_t)$ is the $\Delta C^P(t)$-RSR solution of $S(\tilde{N}_t)$. The global new schedule is $S(t) = S(\hat{N}_t) + S^P(N_t) + S^{PR}(\tilde{N}_t)$; If $|\tilde{N}_t| = 0$, the PR sub-problem with respect to (7) is solved and the solution $S^P(N_t)$ can be obtained. The global new schedule is $S(t) = S(\hat{N}_t) + S^P(N_t)$. Let $t = t+1$. If $t \leqslant l$, go to Step 2, else go to Step 6.

Step 6. The global new schedule S is the last new schedule, *i.e.* $S = S(l)$. Compute the global schedule makespan $M(S)$ and the schedule deviation $D(S)$. For a pair of specified weights (w_D, w_M), the overall objective $J(S)$ defined as (5) can be obtained.

PR sub-problem with respect to (6) or (7) is solved by all-pair-tree search algorithm similar to that in [2].

Lemma 1. In rolling PRDO, $\sum_{i\in\tilde{N}_t} |b_i(t) - b_i(t-1)| \leqslant |\tilde{N}|\Delta C(t)$.

Proof. Because the new schedule $S^{PR}(\tilde{N}_t)$ is obtained through RSR, the schedule deviation of $S^{PR}(\tilde{N}_t)$ to $S(\tilde{N}_t)$, formulated as $\sum_{i\in\tilde{N}_t} |b_i(t) - b_i(t-1)|$, is determined based on the amount of idle time in $S(\tilde{N}_t)$. Assuming that no idle time exists in $S(\tilde{N}_t)$, the match-up delay $\Delta C(t)$ will make the beginning time of each job in $S(\tilde{N}_t)$ delayed by $\Delta C(t)$, then $\sum_{i\in\tilde{N}_t} |b_i(t) - b_i(t-1)| = |\tilde{N}_t|\Delta C(t)$, which is just the worst case behavior and embodies the largest schedule deviation. Therefore, if there is idle time in $S(\tilde{N}_t)$, the schedule deviation must be smaller than $|\tilde{N}_t|\Delta C(t)$ because of idle time being absorbed. Anyway $\sum_{i\in\tilde{N}_t} |b_i(t) - b_i(t-1)| \leqslant |\tilde{N}_t|\Delta C(t)$. □

Theorem 1. In rolling PRDO driven by disruptions, each local objective is consistent with the global objective, whose optimization is realized separately in each PR. The sum of all local objectives is an upper bound for the actual global objective.

Proof. According to the aforementioned PRDO algorithm, if the number of disruptions is l, the global new schedule goes through $S(1), S(2), \cdots$ to $S(l)$ from S^0 during the execution. Since $S(l)$ is exactly the ultimate new schedule S, the schedule deviation of S to S^0 is accumulatively obtained in l times of rescheduling.

$$
\begin{aligned}
D(S) = & \sum_{i\in N} |b_i(l) - b_i(0)| = \\
& \sum_{i\in N} |b_i(l) - b_i(l-1) + b_i(l-1) - b_i(l-2) + b_i(l-2) - \cdots + b_i(1) - b_i(0)| \leqslant \\
& \sum_{i\in N} \{|b_i(l) - b_i(l-1)| + |b_i(l-1) - b_i(l-2)| + \cdots + |b_i(1) - b_i(0)|\} = \\
& \sum_{i\in N}\sum_{t=1}^{l} |b_i(t) - b_i(t-1)| = \sum_{t=1}^{l}\sum_{i\in N} |b_i(t) - b_i(t-1)| = \\
& \sum_{t=1}^{l} \{\sum_{i\in\hat{N}_t} |b_i(t) - b_i(t-1) + \sum_{i\in N_t} |b_i(t) - b_i(t-1) + \sum_{i\in\tilde{N}_t} |b_i(t) - b_i(t-1)|\}
\end{aligned}
$$

Due to Lemma 1

$$D(S) \leqslant \sum_{t=1}^{l}\{\sum_{i\in N_t}|b_i(t)-b_i(t-1)|+|\tilde{N}|\Delta C(t)\}$$

$$J(S) = D(S)+M(S) \leqslant \sum_{t=1}^{l}\{\sum_{i\in N_t}|b_i(t)-b_i(t-1)|+|\tilde{N}|\Delta C(t)\}+M(S) =$$

$$\sum_{t=1}^{l-1}\{\sum_{i\in N_t}|b_i(t)-b_i(t-1)|+|\tilde{N}|\Delta C(t)\}+\{\sum_{i\in N_t}|b_i(t)-b_i(t-1)|+M(S)\}=\sum_{t=1}^{l}J_t$$

The proof procedure as well as the result shows that each local objective is a portion of the global objective and each PR partly optimizes the global objective meanwhile optimizing the local objective. Therefore, each local objective is consistent with the global objective, whose optimization is realized separately in each time of rescheduling. The sum of all local objectives is an upper bound for the global objective. □

The proof of Theorem 1 shows that there are two places where inequality sign appears. We can easily prove that equalities hold respectively at two signs of inequality under the following extreme situation, when the actual global objective reaches the upper bound. We describe the corollary of Theorem 1 as follows and omit the proof.

Corollary. If there is enough idle time in each unimplemented original schedule, the rolling PRDO can make the interim new schedule non-delay match-up its original schedule within each PR horizon and the sum of local objectives is exactly the actual global objective.

The corollary demonstrates that the upper bound in Theorem 1 can be reached and it is a tight upper bound. Under such environment the global objective can be optimized if each local objective is separately optimized in each rescheduling.

Since FR under local search algorithm can merely deal with rescheduling problems with small or medium sizes, it is impossible to compare PRDO with FR in large-size problems. Though RSR simply shifts the unfinished operations to the right without any consideration of objective optimality, we can use an RSR solution as a baseline where our approach is easily examined due to its low computational burden. If RSR is performed instead of PR after each disruption, rolling rescheduling mechanism is referred to as rolling RSR. Comparing rolling PRDO with rolling RSR, we can obtain the following Theorem 2:

Theorem 2. The sum of each local objective in rolling PRDO must be no larger than that in rolling RSR.

5 Computational results and analysis

In the following experiment, all procedures were coded in C language and ran on the Microsoft Visual c++ 6.0 under the Windows XP operating environment. All tests ran on a computer with Pentium 4-M CPU 1.80GHz. Problems were randomly generated using a format similar to that used in [3]. The initial schedule was created by use of Schrage′s algorithm[7]. Three disruptions were generated during a run. The duration of disruptions ranged from five percent to ten percent of the processing time of the initial schedule. We assumed that disruptions would not occur among the last twenty jobs because the number of the rescheduled jobs would be too small to make the rescheduling trivial in those cases. Testing was conducted to compare rolling PRDO with rolling RSR for each problem.

ρ is a range parameter used to control how rapidly jobs are expected to arrive for processing. When ρ value is 0.20, jobs arrive rather rapidly so that almost no idle time exists in the initial schedule. However, when ρ value is 2.00, jobs arrive rather slowly so that much idle time is inserted in the initial schedule. Therefore, the problems with three ρ values actually represent three situations where different amount of idle time exists in the initial schedule. The larger the ρ value is, the more the idle time is. Problems with four sizes of PR horizon 10-job, 20-job, 30-job, 40-job were tested respectively. Each entry was obtained from the statistic results of 20 instances.

5.1 Comparing the sum of local objectives with the global objective

Theorem 1 indicates that the sum of local objectives is an upper bound of the actual global objective. The ratio of the actual global objective to the upper bound reflects the gap between them,

which is affected by many factors. We only explored the effect of the amount of idle time in initial schedule and the size of PR horizon on the ratio.

Table 1 shows the results for 200-job problems. Four hundred problems are totally tested. The percentage ratio of the actual global objective to the sum of local objectives is calculated as $J \Big/ \sum_{t=1}^{l} J_t$. The smaller the percentage ratio is, the larger the gap between the global objective and the upper bound is. The cases where the ratio reached 100% represents zero-gap cases, where the actual global objective reached the upper bound and the sum of local objectives is exactly the actual global objective.

Table 1 The percentage of the actual global performance versus the sum of local objectives

Range parameter (ρ)	0.20			1.00			2.00		
Size of PR horizon (κ)	Ave.	Max.	Min.	Ave.	Max.	Min.	Ave.	Max.	Min.
10	97.1	97.6	96.5	90.1	97.5	60.6	31.9	45.3	24.3
20	94.1	95.2	93.3	87.6	95.0	57.8	47.8	89.3	27.2
30	91.3	92.8	90.3	84.6	92.5	55.1	70.0	100	33.7
40	87.5	89.1	84.8	80.2	89.9	68.4	91.5	100	76.1

Table 1 shows that the gap size is strongly affected by the amount of idle time in the initial schedule as well as the size of PR horizon. If there is less idle time in the initial schedule, the gap is smaller and it will increase as the size of PR horizon gets large. However, no zero-gap case occurs when the ρ value is 0.20 or 1.00. If there is more idle time among the initial schedule, the gap gets larger and it will decrease as the size of PR horizon gets large. When the size of PR horizon is 30 or 40, zero-gap cases occur in $\rho = 2.00$. It demonstrates that the PR horizons are large enough so that enough idle time is accumulated to make each new schedule match-up its original schedule within PR horizons, *i.e.* Corollary of Theorem 1 is testified.

5.2 Comparing the rolling PRDO with the rolling RSR

Testing was conducted to compare rolling PRDO with rolling RSR. The rolling PRDO and the rolling RSR were respectively performed in response to disruptions during the execution. The percentage improvements of rolling PRDO over rolling RSR were calculated as (RSR-PRDO)/PRDO. Tables 2, 3 and 4 show the results for 200-job problems with three ρ values. Total 80 problems were tested.

Table 2 The percentage improvements of rolling PRDO over rolling RSR for 200-job problems: $\rho = 0.20$

Size of PR horizon (κ)	$D(S)$			$M(S)$			$J(S) = D(S) + M(S)$		
	Ave.	Max.	Min.	Ave.	Max.	Min.	Ave.	Max.	Min.
10	1.52	1.97	0.80	0	0	0	1.41	1.85	0.75
20	4.67	6.61	3.04	0	0	0	4.30	6.01	2.87
30	7.97	14.3	5.14	0	0	0	7.33	12.9	4.81
40	12.1	17.3	8.48	0	0	0	10.6	15.6	7.95

Table 3 The percentage improvements of rolling PRDO over rolling RSR for 200-job problems: $\rho = 1.00$

Size of PR horizon (κ)	$D(S)$			$M(S)$			$J(S) = D(S) + M(S)$		
	Ave.	Max.	Min.	Ave.	Max.	Min.	Ave.	Max.	Min.
10	1.39	2.54	0.83	0.04	0.61	-0.21	1.22	2.10	0.74
20	4.98	8.53	2.54	0.99	4.09	-1.36	4.47	7.09	2.37
30	8.37	15.6	4.01	2.08	5.99	-1.08	7.54	12.8	3.74
40	13.2	31.8	7.84	3.43	8.34	0	11.7	24.1	7.26

Table 4 The percentage improvements of rolling PRDO over rolling RSR for 200-job problems: $\rho = 2.00$

Size of PR horizon (κ)	$D(S)$			$M(S)$			$J(S) = D(S) + M(S)$		
	Ave.	Max.	Min.	Ave.	Max.	Min.	Ave.	Max.	Min.
10	4.95	9.37	1.62	0.06	0.47	0	2.48	4.07	1.05
20	13.1	20.6	8.28	0.10	1.11	0	6.73	10.3	3.37
30	17.1	26.1	7.52	0.07	1.38	0	8.82	14.6	3.38
40	21.4	30.5	12.3	0.05	0.52	0	11.3	15.9	4.79

It is obviously shown that the schedule stability of rolling PRDO is largely improved over that of RSR. Though the improvements of schedule efficiency is trivial in most cases and even decline a little in some other cases, the overall objective for rolling PRDO is obviously improved over that for rolling

RSR when we pay equal weights for the dual objectives. The improvements get larger as the size of PR horizon increases. The computational results also indicate that the improvements of stability are larger when more idle time exists in the initial schedule.

Fig. 1 presents CPU time paid by rolling PRDO under different sizes of PR horizon. It is shown that more CPU time should be paid for larger improvements achieved by rolling PRDO with larger PR horizon. The rolling PRDO can provide a reasonable trade-off between solution quality and computational efforts.

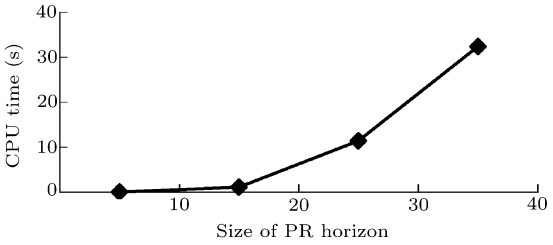

Fig. 1 CPU time of rolling PRDO for 200-job problems with $\rho = 1.00$

6 Conclusions

Aiming at large-scale dynamic production planning environments, rolling partial rescheduling is developed in response to more disruptions. The new schedule is required to satisfy dual objectives: efficiency and stability. Two kinds of PR objective function, where the global dual objectives are reflected to some extent, are respectively designed for the interim and the terminal of PR horizon. The correlation of local objectives to the global objective is theoretically analyzed. The analytical conclusions demonstrate that each local objective is consistent with the global one and the sum of local objectives is an upper bound of the actual global objective. Extensive computational experiment has been performed and the computational results show that the rolling PRDO can greatly improve schedule stability with little sacrifice in efficiency and provide a reasonable trade-off between solution quality and computational efforts. The rolling partial rescheduling is effective for large-scale dynamic rescheduling problems with more disruptions.

References

1 Bayiz S M. Analysis of reactive scheduling problems in a job shop environment. *European Journal of Operational Research*, 2000, **126**: 567∼586
2 Wu D S, Storer R H, Chang P C. One-machine rescheduling heuristics with efficiency and stability as criteria. *Computers in Operations Research*, 1993, **20**(1): 1∼14
3 Wang B, Xi Y G, Gu H Y. Terminal penalty rolling scheduling based on an initial schedule for single-machine scheduling problem. *Computers and Operations Research*, 2005, **32**(11): 3059∼3072
4 Wang B, Xi Y G, Gu H Y. An improved rolling horizon procedure for single-machine scheduling with release times. *Control and Decision*, 2005, **20**(3): 257∼260
5 Garey M R, Johnson D S. Computers Intractability. Freeman, San Francisico, Calif., 1979
6 Bean J C, Birge J R, Mittenehal J, Noon C E. Match-up scheduling with multiple resources, release dates and disruption. *Operations Research*, 1991, **39**(3): 470∼483
7 Carlier J. The one-machine sequencing problem. *European Journal of Operational Research*, 1982, **11**: 42∼47

WANG Bing Associate professor of Shandong University at Weihai. Received her Ph. D. degree from Shanghai Jiaotong University in 2005. Her research interests include production scheduling and combinatorial optimization.

XI Yu-Geng Professor of Shanghai Jiaotong University. Received his Ph. D. degree from Technical University of Munich, Germany in 1984. His research interests include predictive control, large-scale system, and intelligent robotics.