Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Scheduling (CI-Sched 2007)

Rolling Partial Rescheduling Driven by Disruptions on Single-machine Based on Genetic Algorithm

Bing Wang, and Xiaoying Hong

Abstract—**This paper discusses large-scale single-machine rescheduling problems with efficiency and stability as bi-criterion, where more than one disruption arises during the execution of an initial schedule. Partial rescheduling (PR), which involves only partial unfinished schedules, is adopted in response to each disruption and forms a PR sub-problem. The remaining unfinished schedule is just right-shifted or not following the solution of PR sub-problem. During the process of schedule execution, a rolling PR strategy is driven by disruption events. Each global rescheduling consisting of two segments of local rescheduling revises the original schedule into a new schedule, which is exactly the next original schedule. Two types of local objective functions are designed for PR sub-problems locating in the process or the terminal of original schedules respectively, where the global information of bi-criterion problems is reflected to an extent. The analytical results demonstrate that each local PR objective is consistent to the global one. For PR sub-problems with such a particular criteria, a genetic algorithm is used to solve it. Extensive computational experiments were performed. Computational results show that the rolling PR can greatly improve schedule stability with a little sacrifice in schedule efficiency and consistently outperforms the rolling right-shift rescheduling. The rolling PR strategy is effective to address large-scale rescheduling problems with more disruptions.**

I. Introduction

A deterministic initial schedule is 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 disturbances, such as rush orders, excess processing time, and machine breakdown etc, will arise during the execution of such an initial schedule. Rescheduling is usually performed to deal with such dynamic environments. A new schedule can be obtained by revising the unfinished schedule on occurrence of disruptions.

Reactive rescheduling can be performed timely driven by disruption events or periodically [1]. Two types of reactive rescheduling strategies were used in the existing literatures. Full rescheduling (FR), where all unfinished jobs are rescheduled to satisfy certain objective, can be merely applied in problems with small or medium size though it can result in an optimal solution. Right-shift rescheduling (RSR), where all unfinished jobs are just slid to the right as far as necessary to accommodate (absorbing idle time) the disruption, cannot guarantee the solution quality though it requires less computational efforts. Compromising FR and RSR, partial rescheduling can provide a trade-off between solution quality and computational cost through considering only partially unfinished jobs.

The practical solution of rescheduling problems requires satisfaction of two often conflicting goals: (1) to retain schedule efficiency, i.e. to keep the schedule performance less degraded as possible as we can, and (2) simultaneously to minimize the cost impact of the schedule deviation. Wu et al. [2] addressed rescheduling problems with efficiency and stability, where only one disruption occurs. Sabuncuoglu et al. [1] used a PR strategy to deal with the large-scale rescheduling problems. However, the schedule stability is often poor because of merely considering the schedule efficiency.

During the execution of a large-scale initial schedule, more than one disruption possibly arises due to the long execution duration. Furthermore, the number of unfinished operations at one disruption is likely too large to reschedule. Following the experiences of Wang et al. addressing large-scale scheduling problems in [3] and [4], we develop a rolling PR strategy to deal with large-scale rescheduling problems with efficiency and stability for single machine on occurrence of more disruptions in this paper.

II. 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, which represents the processing time on subsequent machines in the system. These three aforementioned parameters of each job are known a priori. For a solution S for this problem, the makespan, denoted as $M(S)$, is defined as follows:

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

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

Manuscript received October 29, 2006. This work was supported partly by Science Research Foundation of Shandong University at Weihai (XZ2005001)

Bing Wang (corresponding author) is with the Department of Automation, School of Information Engineering, Shandong University at Weihai, Shandong 264209, People's Republic of China. (Telephone number: 0086-0631-2982649; fax: 0086-0631-5688338; e-mail: wangbing@sdu.edu.cn).

Xiaoying Hong is with the Department of Automation, School of Information Engineering, Shandong University at Weihai, Shandong 264209, People's Republic of China. (e-mail: snowbabyhong@sohu.com)

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 should be rescheduled. The release times of all unfinished jobs are updated as follows:

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

In this paper, the schedule stability, denoted as $D(S)$, is measured by the schedule deviation based on initial schedule, 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 S^0.

The rescheduling problems with efficiency and stability are to minimize both (1) and (3). An optimization problem with such two goals can be formulated to be a problem with the following objective

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

This problem is also NP-hard [2].

III. PR SUB-PROBLEM AT EACH DISRUPTION

In large-scale dynamic circumstances with more disruptions, FR strategy is neither beneficial nor needed because many operations are probably rescheduled more than one time. RSR strategy does not take any consideration of objective optimality. Therefore, PR strategy will be a better choice due to its compromise between FR and RSR.

In this paper, we use t to represent the rescheduling moment driven by a disruption. At t, the original schedule refers to the new schedule obtained through rescheduling at the previous disruption. Obviously, the original schedule for the first rescheduling, when $t = 1$, is exactly the initial schedule. The first rescheduling revises the first original schedule into the first new schedule, which will be implemented until the next disruption occurs. In a general way, the original schedule at t is the new schedule at $t-1$, denoted as $S(t-1)$.

In the PR strategy of the following sections, the global rescheduling consists of two segments of local rescheduling. At each disruption, from the interrupted jobs on, all unfinished jobs are divided into two portions. The first portion from the beginning jobs of the original schedule, which forms a PR sub-problem, is totally rescheduled with respect to a certain criteria, whereas for the jobs of the remaining portion, RSR is performed following the solution of the PR sub-problem. In such a rescheduling pattern, the global new schedule after each disruption consists of the PR solution and the RSR solution.

Let N be the set of all jobs. Let $\underset{\sim}{N}_t$ be the set of finished jobs at t and $\underset{\sim}{S}(t)$ be the partial original schedule for $\underset{\sim}{N}_t$. Let N_t' be the set of the unfinished jobs at t (In this paper, we assume that the interrupted job should be resumed in rescheduling and included into N_t'), then $N = \underset{\sim}{N}_t \cup N_t'$.

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

Definition 2. Let the initial schedule for a single-machine scheduling problem be S^0, the RSR solution S^R to S^0 is referred to as Δt -RSR solution to S^0 if the first unfinished job is shifted to right by a time interval Δt.

Let $\tilde{N}_t$ be the set of the remaining jobs in N_t' except N_t, i.e. $N_t' = N_t \cup \tilde{N}_t$. The number of jobs in $\tilde{N}_t$ is denoted as $|\tilde{N}_t|$. The global rescheduling involves the PR for N_t and the RSR for $\tilde{N}_t$. At t, the new schedule consists of the local new schedule for N_t and the latter new schedule for $\tilde{N}_t$.

The ideas of match-up rescheduling (MUR) came from Bean and Birge [6]-[7]. When a disruption occurs, rescheduling involving a transitional period of the initial schedule can accommodate the disruption and make the new schedule completely consistent to the initial schedule from the terminal point of the transitional schedule on. Rescheduling with respect to such goal is referred to as MUR. The terminal point of the transitional schedule is referred to as a match-up point and the transitional period is referred to as a match-up time. Suppose that a match-up point was forced on the initial schedule in advance, a new schedule would match up the initial one 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 for the match-up point. In the following, we present the definition of "match-up delay" and apply it in the design of criteria for PR sub-problems.

Definition 3. For a forced match-up point of initial schedule, a time interval Δt is referred to as a match-up delay if the completion time of match-up point is delayed by Δt in the new schedule. When $\Delta t = 0$, there is a non-delay match-up of the new schedule to the initial schedule.

At t, when a disruption occurs in the unfinished original schedule $S(t-1)$, from the interrupted point on, the partial schedule based on PR-horizon N_t is denoted as $S(N_t)$. In order to implement the global PR strategy, the completion time of $S(N_t)$, which is denoted as $C(t-1)$, would be regarded as a match-up point forced in $S(t-1)$. After rescheduling, the new schedule for N_t is denoted as $S^p(N_t)$, whose completion time is denoted as $C^p(t)$. If $C^p(t) > C(t-1)$, the match-up delay in N_t is $\Delta C^p(t) = C^p(t) - C(t-1)$. Else if $C^p(t) \le C(t-1)$, the match-up delay $\Delta C^p(t) = 0$, i.e. a non-delay match-up of

new schedule to the original schedule can be obtained in N_t. Let the $\Delta C^p(t)$-RSR solution to $S(t-1)$ for $\tilde{N}_t$ be $S^{pr}(\tilde{N}_t)$. Thus, the new schedule $S(t)$ consists of $S^p(N_t)$ and $S^{pr}(\tilde{N}_t)$.

Two types of local objective functions for PR sub-problems are defined as follows based on PR-horizon N_t locating in the process and the terminal of original schedules respectively:

$$\min_{S^p(N_t)} J_t = \{\sum_{i\in N_t} | b_i^p(t) - b_i(t-1) | + | \tilde{N}_t | [\Delta C^p(t)]\} \quad | \tilde{N}_t | > 0 \quad (5)$$

$$\min_{S^p(N_t)} J_t = \{\sum_{i\in N_t} | b_i^p(t) - b_i(t-1) | + M(S(t))\} \quad | \tilde{N}_t | = 0 \quad (6)$$

Where $b_i(t-1)$ denotes the beginning time of job i in $S(t-1)$ and $b_i^p(t)$ denotes the beginning time of job i in $S^p(N_t)$. (5) is designed for N_t locating in the process of original schedules. The local objective of PR sub-problems is to minimize both the schedule deviation and the match-up delay $\Delta C^p(t)$ for N_t. Since the consideration of the match-up delay in PR-horizon would make the new schedule inserted by less idle time, it is reasonable to use the number of latter jobs as the weight for the match-up delay in case more idle time greatly puts off the latter jobs. In fact, the second item is exactly an upper bound for the schedule deviation in $\tilde{N}_t$ when the match-up delay for N_t is $\Delta C^p(t)$. In such a pattern, the schedule deviation for $\tilde{N}_t$ happens to be considered in the PR local rescheduling and it results in that the global information is considered to an extent in local PR. Such an elaborately designed criteria will make the PR strategy more effective.

When N_t locates in the terminal of original schedules, the set $\tilde{N}_t$ is empty and we directly consider the makespan in PR sub-problems according to (6).

IV. Rolling PR based genetic algorithm

When disruptions occur during the schedule execution, PR is driven by disruption events in a rolling mechanism. Let x be the number of disruptions, the rolling PR is performed as follows:

Step 1 Minimize the makespan to generate the initial schedule S^0 without considering any disruption, let $t=1$, then the first original schedule $S(0)=S^0$;

Step 2 Implement the original schedule $S(t-1)$ until a disruption occurs, when the moment 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' are updated according to (2);

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

Step 5 If $|\tilde{N}_t| > 0$, the PR sub-problem is formed according to (5). Its solution $S^p(N_t)$, the completion time of $S^p(N_t)$, $C^p(t)$ as well as the match-up delay $\Delta C^p(t)$ in N_t can be obtained. Let $S^{pr}(\tilde{N}_t)$ be the $\Delta C^p(t)$-RSR solution for $\tilde{N}_t$ to the original schedule. Thus the new schedule is $S(t) = S(\underset{\sim}{N}_t) + S^p(N_t) + S^{pr}(\tilde{N}_t)$; Else if $|\tilde{N}_t| = 0$, the PR sub-problem is formed according to (6) and the solution $S^p(N_t)$ can be obtained. Thus the new schedule is $S(t) = S(\underset{\sim}{N}_t) + S^p(N_t)$. Let $t = t+1$. If $t \le x$, go to Step 2, else go to Step 6;

Step 6 The global new schedule S is the last new schedule, i.e. $S = S(x)$. Calculate the global schedule makespan $M(S)$ and the schedule deviation $D(S)$, the objective $J(S)$ defined as (4) can be obtained;

In this paper, since the size of PR sub-problems is restricted, a genetic algorithm can be used to solve sub-problems formulated by (5) or (6). We can encode the schedules and generate the initial population in the similar manners to Wu et al. [2]. However, since our objectives differ from those of Wu et al., the genetic algorithm in [2] should be modified.

For the bi-criterion scheduling problem, Wu et al. encode by converting the sequence into a string of "artificial tails". Let $j = 1, \cdots, n$ index jobs in N_t ordered according to the sequence to be encoded. An artificial tail is defined as follows: $q_k' = 0$, $q_{j-1}' = q_j' + p_j$.

Thus the earlier a job appears in the sequence, the longer its artificial tail will be. A list, $\{q_1', \cdots, q_k'\}$, ordered by the job number is maintained for each solution as its "chromosome". This encoding can be manipulated by a typical crossover operator, and can be evaluated by applying Schrage's heuristic [8], which guarantees schedule feasibility.

To start the genetic algorithm, an initial solution population must be generated. Two desirable properties of the initial population are (1) high quality solutions (corresponding to fitness), and (2) diverse solutions. A heuristic method developed by Wu et al. is used to generate the initial population. The method for initial population

generation uses the solutions produced by a fast, parametrically varied heuristic termed "$\alpha-\varepsilon$ grid search", which allow varying emphasis to be placed on the two criteria.

Once the initial population is generated, each solution in the population is encoded by the string of artificial tails, $Q_y=\{q'_1,\cdots,q'_k\}$, used to generate the schedule. From the current population, P, a number of "more fit" solutions will be selected for reproduction based on the fitness measure.

A critical element of a genetic algorithm is the measure of solution fitness. The fitness measure represents the selection probability of a solution during the search. For equation (5), the fitness measure for a given PR solution, $S^y(N_t)$, is then defined as follows:

$$f_y=[J_{t\max}-J_{ty}]^l\Big/\sum_{j\in P}[J_{t\max}-J_{tj}]^l$$

Where

$$J_{t\max}=\sum_{i\in S^c(N_t)}|b_i^c(t)-b_i(t-1)|+|\tilde{N}_t|[C^m(t)-C(t-1)],$$

$$J_{ty}=\sum_{i\in S^y(N_t)}|b_i^y(t)-b_i(t-1)|+|\tilde{N}_t|[C^y(t)-C(t-1)].$$

Where $S^c(N_t)$ is the PR solution by Carlier's algorithm [8], $b_i^c(t)$ and $b_i^y(t)$ are the beginning times of job i in $S^c(N_t)$ and $S^y(N_t)$. $C^m(t)$ is the completion time of the minimum deviation PR solution, $C^y(t)$ is the completion time of $S^y(N_t)$. l is a constant used for tuning purposes. As l increases, the genetic algorithm becomes more selective when generating off-spring solutions. With a large l-value, only the "most fit" solution will survive and the algorithm converges prematurely in one step.

The genetic algorithm can be summarized as follows:

Step 1 Initialization: start with an initial set of tuning parameters [i.e. l, population size (*ps*), mutation probability (*mp*), and number of generations (*ng*)], generate an initial population by "$\alpha-\varepsilon$ grid search", compute the fitness f_y, $\forall S^y(N_t)\in P$, and save the set of artificial tails, Q_y, corresponding to each solution $S^y(N_t)$.

Step 2 $g=1$, start iterations.

Step 3 Copy the set of *mb* best solutions from the current population to the next generation.

Step 4 Perform crossover $(ps-mb)/2$ times as follows: select a distinct pair of solutions $(S^y(N_t),S^j(N_t))$ randomly from the current population based on probabilities: f_y and f_j; (b) generate two crossover sites x_1 and x_2 $(x_1<x_2)$ from a discrete uniform distribution between 1 and $|N_t|$; (c) swap the sublist $\{q'_{y1},\cdots,q'_{yk}\}$ of Q_y with the corresponding sublist in $Q_j=\{q'_{j1},\cdots,q'_{jk}\}$, apply Schrage's heuristic to obtain the new schedule, compute the criteria J_{ty}, J_{tj}; (d) copy both off-spring solution to the next generation, compute f_y, f_j, and save Q_y, Q_j.

Step 5 For each solution $S^y(N_t)$, apply mutation with probability, *mp*. Mutation is performed by setting $\alpha=uniform[0,1]$, then recomputed all the artificial tails in Q_y by $q''_i=(1-\alpha)q'_i+\alpha q_i$.

Step 6 Set the generation as the current population, $g=g+1$, if $g<ng$, go to step 3, else stop.

V. Analysis of Rolling PR

Lemma. In the rolling PR, for a match-up delay $\Delta C(t)$ in N_t, $\sum_{i\in\tilde{N}_t}|b_i(t)-b_i(t-1)|\le|\tilde{N}_t|\Delta C(t)$.

Proof. Assumed that no idle time exists in the original schedule for $\tilde{N}_t$, the match-up delay $\Delta C(t)$ will make the beginning time of each job delayed by $\Delta C(t)$, then the schedule deviation for $\tilde{N}_t$ between the new schedule and the original schedule, which is formulated as $\sum_{i\in\tilde{N}_t}|b_i(t)-b_i(t-1)|$, is exactly $|\tilde{N}_t|\Delta C(t)$. This situation embodies the largest schedule deviation among all situations and it is the worst case. Therefore, if there is idle time in the original schedule for $\tilde{N}_t$, the schedule deviation must be less than $|\tilde{N}_t|\Delta C(t)$ due to idle time being absorbed. Anyway, $\sum_{i\in\tilde{N}_t}|b_i(t)-b_i(t-1)|\le|\tilde{N}_t|\Delta C(t)$. □

Following the aforementioned rolling PR, if the number of disruptions is x, the global new schedule goes through $S(1),S(2),\cdots S(x)$ from S^0 during the execution. Since $S(x)$ is exactly the ultimate new schedule S, the schedule deviation between S and S^0 is accumulatively realized in x times of local rescheduling. Therefore, we have the following Theorem 1.

Theorem 1. In rolling PR driven by more disruptions, each local PR objective is consistent with the global objective, whose optimization is realized separately in each local rescheduling. The sum of local PR objectives is an upper bound for the global one, i.e.

$$J(S)=D(S)+M(S)\le\sum_{t=1}^{x}J_t$$

Proof.

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

Due to Lemma

$$
\begin{aligned}
D(S) &\le \sum_{t=1}^{x} \{ \sum_{i\in N_t} |\, b_i(t) - b_i(t-1) \,| + |\, \tilde{N}_t \,|\, \Delta C(t) \} \\
J(S) = D(S) + M(S) &\le \sum_{t=1}^{x} \{ \sum_{i\in N_t} |\, b_i(t) - b_i(t-1) \,| + |\, \tilde{N}_t \,|\, \Delta C(t) \} + M(S) \\
&= \sum_{t=1}^{x-1} \{ \sum_{i\in N_t} |\, b_i(t) - b_i(t-1) \,| + |\, \tilde{N}_t \,|\, \Delta C(t) \} + \{ \sum_{i\in N_l} |\, b_i(x) - b_i(x-1) \,| + M(S) \} \\
&= \sum_{t=1}^{x} J_t
\end{aligned}
$$

□

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

From the conclusions in [2], we can easily obtain the following theorem.

Theorem 2. After a local PR, a non-delay match-up of new schedule to original schedule can be obtained as long as enough idle time exists in the original schedule for PR-horizon.

The global objective can be optimized if each local objective is separately optimized in each local PR. We describe the conclusions as follows:

Corollary 1. If there is enough idle time in each unfinished original schedule, the rolling PR can make each new schedule non-delay match-up its original schedule within each PR-horizon and the sum of local objectives is exactly the global objective.

Corollary 1 demonstrates that the upper bound in Theorem 1 can be reached and it is a tight upper bound.

If RSR is performed instead of PR at each disruption, such rolling rescheduling is referred to as rolling RSR. Comparing rolling PR with rolling RSR, we can obtain the following corollary:

Corollary 2. The sum of local objectives in rolling PR is no more than that in rolling RSR.

VI. COMPUTATIONAL RESULTS AND ANALYSIS

In tests of this section, the initial schedule was generated by use of Schrage's algorithm [8]. All procedures were coded in C language and ran in 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. Three disruptions would occur during a run. The durations of disruption range from five percent to ten percent of the processing time of the initial schedule. We assumed that the disruption would not occur among the last twenty jobs because the number of the rescheduled jobs is too few to make rescheduling trivial in those cases.

Problems were randomly generated using a format similar to that used by Ovacik et al. [9], where the range parameter ρ is 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, the situation when ρ value is 2.00 is the reverse, 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 schedules.

In the genetic algorithm for PR sub-problem, we used $l = 2$, population size $ps = 100$, mutation probability $mp = 0.05$, and the number of generations $ng = 200$. When the $\alpha - \varepsilon$ procedure was used to generate initial population, $\Delta\alpha = 0.01$ and $\Delta\varepsilon = 0.5$.

A. Testifying the conclusions for rolling PR

Theorem 1 indicates that the sum of local objectives is an upper bound of the global objective. The gap between the upper bound and the actual global objective is affected by many factors, among which we only examined the effect of

the amount of idle time in initial schedule and the size of PR-horizon.

Table 1 shows the results for 200-job problems. Three ρ values represent respectively three situations of idle time amount among the initial schedule: the larger the value of ρ is, the more the idle time is. The PR-horizon size was specified to be total four types, which is 10-job, 20-job, 30-job, and 40-job. Each entry was obtained from the statistic results of 20 instances. 400 problems are tested. The percentage ratio of the actual global objective to the sum of local objectives is calculated as $100 \times J \Big/ \sum_{t=1}^{x} J_t$. 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, i.e. the actual global objective reached the upper bound and the sum of local objectives is exactly the actual global one.

The computational results of Table 1 indicate that the gap between the upper bound and the global objective is strongly affected by the idle time in the initial schedule as well as the PR-horizon size. If there is less idle time among the initial schedule, the gap is smaller and will be increased as the PR-horizon size gets large. When the PR-horizon size is 30 or 40 and ρ is large, zero-gap can be reached, which demonstrates that the PR-horizons are large enough so that enough idle time is accumulated to make each new schedule non-delay match-up its original schedule in PR-horizons, i.e. the conditions of Corollary 1 can be satisfied.

B. Comparing rolling PR with rolling RSR

We can use a RSR solution as a baseline where our approach is compared due to its low computational burden. The rolling PR and the rolling RSR were respectively performed in response to disruptions during the execution of initial schedule. The percentage improvements of rolling PR over rolling RSR were calculated as $(RSR - PR)/PR$. Table 2, 3 and 4 show the results of 200-job problems with three ρ values. Each entry was obtained from the statistic results of 20 problems.

It is obviously shown that the schedule stability for rolling PR was largely improved over that for rolling RSR. Though the improvements of schedule efficiency were trivial in most cases and even declined a little in some cases, the overall objective for rolling PR was obviously improved over that for rolling RSR. The improvements were obviously getting larger as PR-horizon size increases. The computational results also indicate that the improvements of schedule stability are more when more idle time exists in the initial schedule.

Figure 1 presents CPU time paid by rolling PR with different PR-horizon sizes. It indicates that more CPU time should be paid for more improvements achieved by rolling PR with larger PR-horizon.

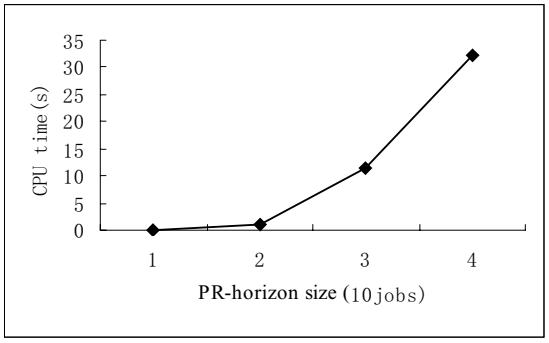

Fig. 1. CPU time of rolling PR for 200-job problems with 1.00 range parameter

VII. CONCLUSIONS

Aiming at large-scale rescheduling problems with disruptions, the rolling partial rescheduling strategy driven by disruption events is adopted in this paper. The new schedule is required to satisfy two goals: efficiency and stability. Two particular types of PR sub-problems are respectively designed for PR-horizon locating in the process and the terminal of original schedules. A genetic algorithm is used to solve PR sub-problems. The relation between local PR objectives and the global objective is analyzed and conclude that the sum of local PR objectives is an upper bound for the global one. Extensive computational experiments were performed. The computational results show that the rolling PR can greatly improve the schedule stability with a little sacrifice in schedule efficiency and consistently outperforms the rolling RSR. The rolling PR based on genetic algorithm is effective for large-scale rescheduling problems with more than one disruption.

REFERENCES

[1] S. M. Bayiz, "Analysis of reactive scheduling problems in a job shop environment," European Journal of Operational Research, 2000, 126: 567-586.

[2] D. S. Wu, R. H. Storer, and P. C. Chang, "One-machine rescheduling heuristics with efficiency and stability as criteria," Computers in Operations Research, 1993, 20(1): 1-14.

[3] B Wang, Y. G. Xi, and H. Y. Gu, "An improved rolling horizon procedure for single-machine scheduling with release times," Control and Decision，2005, 20(3): 257-260

[4] B Wang, Y. G. Xi, and H. Y. Gu, "Terminal penalty rolling scheduling based on an initial schedule for single-machine scheduling problem," Computers and Operations Research, 2005, 32(11): 3059-3072.

[5] M. R. Garey, D. S. Johnson, "Computers Intractability", Freeman, San Francisico, Calif., 1979.

[6] J. C. Bean, J. R. Birge, "Match-up real-time scheduling," Proceedings of the Symposium on Real Time Optimization in Automated Manufacturing Facilities, NBS publication 724, National Bureau of Standards, 1985: 197-212.

[7] J. C. Bean, J. R. Birge, J Mittenehal, and C. E. Noon, "Match-up scheduling with multiple resources, release dates and disruption," Operations Research, 1991, 39(3): 470-483.

[8] J. Carlier, "The one-machine sequencing problem," European Journal of Operational Research, 1982, 11: 42-47.

[9] I. M. Ovacik, R. Uzsoy, "Rolling horizon algorithms for a single-machine dynamic scheduling problem with sequence-dependent setup times," International Journal of Production Research, 1994, 32(6): 1243-1263.

TABLE 1
THE PERCENTAGE OF THE ACTUAL GLOBAL PERFORMANCE VERSUS THE SUM OF LOCAL PR OBJECTIVES

Range parameter (ρ) / PR-horizon Size (κ)	0.20			1.00			2.00		
	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 2
THE PERCENTAGE IMPROVEMENTS OF ROLLING PR OVER ROLLING RSR FOR 200-JOB PROBLEMS: $\rho = 0.20$

PR-horizon Size (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 PR OVER ROLLING RSR FOR 200-JOB PROBLEMS: $\rho = 1.00$

PR-horizon Size (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 PR OVER ROLLING RSR FOR 200-JOB PROBLEMS: $\rho = 2.00$

PR-horizon Size (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