J Sched (2010) 13: 95–110
DOI 10.1007/s10951-009-0111-2

Parallel machine match-up scheduling with manufacturing cost considerations

M. Selim Aktürk · Alper Atamtürk · Sinan Gürel

Received: 1 February 2008 / Accepted: 4 May 2009 / Published online: 2 June 2009

Abstract Many scheduling problems in practice involve rescheduling of disrupted schedules. In this study, we show that in contrast to fixed processing times, if we have the flexibility to control the processing times of the jobs, we can generate alternative reactive schedules considering the manufacturing cost implications in response to disruptions. We consider a non-identical parallel machining environment where processing times of the jobs are compressible at a certain manufacturing cost, which is a convex function of the compression on the processing time. In rescheduling it is highly desirable to catch up the original schedule as soon as possible by reassigning the jobs to the machines and compressing their processing times. On the other hand, one must also keep the manufacturing cost due to compression of the jobs low. Thus, one is faced with a tradeoff between match-up time and manufacturing cost criteria.

We introduce alternative match-up scheduling problems for finding schedules on the efficient frontier of this time/cost tradeoff. We employ the recent advances in conic mixed-integer programming to model these problems effectively. We further provide a fast heuristic algorithm driven by dual prices of convex subproblems for generating approximate efficient schedules.

M.S. Aktürk (✉) · S. Gürel
Department of Industrial Engineering, Bilkent University,
06800 Bilkent, Ankara, Turkey
e-mail: akturk@bilkent.edu.tr

S. Gürel
e-mail: sgurel@bilkent.edu.tr

A. Atamtürk
Industrial Engineering & Operations Research Department,
University of California, Berkeley, CA 94720-1777, USA
e-mail: atamturk@berkeley.edu

Keywords Parallel machine rescheduling · Match up times · Controllable processing times · Second-order cone programming

1 Introduction

Expediting jobs by compressing processing times is a critical aspect in making reactive decisions against unexpected disruptions to a schedule. Considering processing time compression decisions at the same time with scheduling decisions, such as sequencing or allocation, enables one to generate alternative schedules with varying manufacturing cost and scheduling performance, and hence brings flexibility in reactive scheduling. In this study, we present how rescheduling and processing time decisions can be made at the same time to react against a machine breakdown on a given schedule.

Given a machine breakdown on a schedule being executed, match-up scheduling aims at finding a new schedule which matches up with the preschedule at some point in the future, called the match-up time. At the match-up time, the new schedule is in the same state as the preschedule. Most of the rescheduling studies in the literature, including the match-up scheduling approach, assume planned idle time periods in preschedules so that a disruption can be absorbed. This assumption simplifies the rescheduling problem, but it reduces the utilization of the expensive machines for a disruption that may or may not occur during the given planning period. With controllable processing times, even if the preschedule is a non-delay schedule, i.e., no idle time exists in the schedule, we can still reschedule to match up with the preschedule. It may be possible to match up soon after the disruption by compressing the jobs which immediately succeed the breakdown. However, catching up sooner implies

incurring more compression cost. Therefore, considering the match-up time and the manufacturing cost objectives at the same time, it is critical to make appropriate rescheduling and processing time decisions.

A well known example for controllable processing times is the CNC turning operation where processing times can be controlled via setting machining parameters such as machining speed and feed rate. Decreasing the processing time of an operation requires incurring additional compression cost. In the CNC machining example, this additional cost is due to higher tooling cost as a result of increased cutting speed and feed rate. Compression cost is a convex function of the amount of compression on the processing time. Nonlinearity of the cost function makes the rescheduling problem harder to solve by using commercial solvers. In this paper, we also show how recent advances in conic quadratic programming can be used to overcome this difficulty.

A predictive approach in coping with disruptions is to leave idle time periods in schedules so that any unexpected event, such as breakdown or new job arrival, can be handled without much disruption. Mehta and Uzsoy (1998) were the first to propose including inserted idle times in a job shop schedule so as to reduce the impact of disruptions. They first find a job sequence and then apply a heuristic approach to insert idle times into the schedule. For the maximum tardiness problem, Jensen (2001) proposed minimizing maximum lateness instead, so that the achieved schedule has improved rescheduling performance due to the idle time left at the end of the schedule. In a recent work, Leus and Herroelen (2007) considered minimizing expected weighted deviation between actual and planned job starting times in a single machine scheduling problem with common deadline for all jobs. They find the optimal job sequence and the optimal length of idle time following each job in the schedule when exactly one job deviates from its expected duration. However, when the processing times are controllable, inserting idle times into the schedules requires selecting smaller processing times which causes higher manufacturing cost for the schedule. If no disruption occurs, the machines will be under-utilized.

For a survey of different approaches in the rescheduling literature we refer the reader to a Vieira et al. (2003). Briand et al. (2007) give an approach to characterize the set of optimal solutions to the single machine maximum lateness problem. Their approach is also used to characterize set of optimal solutions with a worst case performance bound under some execution scenarios regarding the release times and due dates. In the literature, there exist few match-up scheduling studies such as Bean et al. (1991) and Aktürk and Görgülü (1999), which consider heuristic approaches to find match-up times. It is important to note that in flow shop scheduling problems (such as Aktürk and Görgülü 1999) there might be an idle time in the final schedule due to the differences in operational processing times and the precedence relationships.

Scheduling problems with controllable processing times have received significant interest in the literature. Shabtay and Steiner (2007) give an extensive survey on the topic. Gürel and Aktürk (2007) consider minimizing total manufacturing cost subject to a given bound on the makespan objective in non-identical parallel CNC machine environment. Yedidsion et al. (2007) study a single machine scheduling problem to minimize maximum lateness and resource consumption simultaneously. To the best of our knowledge, controllable processing times have not been considered in match-up scheduling problems before.

In this study, we present match-up scheduling problems with controllable processing times which demonstrate the tradeoff between match-up time and manufacturing cost objectives. We give exact and heuristic solution approaches for the considered problems. We propose alternative rescheduling approaches with controllable processing times on non-identical parallel machines. We consider different rescheduling objectives to minimize. The first one is the total manufacturing cost for the jobs not yet started at the time of the disruption. The second objective is the sum of match-up times on the machines. The third one is the maximum match-up time for the new schedule. We consider finding efficient solutions to problems having two objectives to be minimized. One of them is the total manufacturing cost and the other one is either the sum of match-up times or the maximum of match-up times. An efficient solution is the one for which improvement in one objective is not possible without degrading the other objective. Hence, we solve the problem of minimizing total manufacturing cost subject to a constraint on total match-up time or on maximum match-up time. We give the formulations for each of these problems. We show that cost minimization problems can be reformulated by using conic quadratic inequalities. This reformulation is important since it allows us to solve the practical size problems very fast. Solving problems quickly is critical in rescheduling because the schedule continues to be processed while schedule modifications are being determined. The second approach is devising an algorithm which generates a set of approximately efficient solutions for the cost and match-up time objectives based on the gradient of the convex cost function. The notation used throughout the paper is stated below:

Decision Variables:

x_{ij}: 1, if job j is assigned on machine i; 0, otherwise.

z_j: 1, if start time of job j in $\mathcal{S}$ is selected as match-up time; 0, otherwise.

y_{ij}: Compression on the processing time of job j on machine i.

Parameters:

p_{ij}^u: Processing time upper bound that gives the minimum manufacturing cost for job j on machine i.
c_{ij}: Manufacturing cost of job j on machine i with no compression.
u_{ij}: Maximum possible compression for job j on machine i.
$f_{ij}(y_{ij})$: Compression cost function for job j on machine i.
D_i: Available machining time capacity on machine i.
$\mathcal{S}$: Preschedule on which a breakdown occurs.
$y_{ij}^{\mathcal{S}}$: Compression on the processing time of job j on machine i in $\mathcal{S}$.
s_j: Start time of job j in $\mathcal{S}$.
M^*: Index of the disrupted machine.
t^*: Time of disruption on machine M^*.
d^*: Duration of disruption on machine M^*.
J: Set of jobs not yet started processing at the time of disruption, i.e., $J = \{j | s_j \geq t^*\}$.
J_i: Subset of J scheduled on machine i in $\mathcal{S}$.
J_i^m: Subset of J_i that can form match-up point on machine i, i.e., $J_i^m = \{j \in J_i | s_j \geq t^* + d^*\}$ for $i = M^*$ and $J_i^m = \{j \in J_i | s_j \geq t^*\}$ for $i \neq M^*$.
P_j: Set of jobs including job j and its predecessors which can form match-up point on the same machine in $\mathcal{S}$.
E_i: End time of the last job on machine i in $\mathcal{S}$.

In Sect. 2, we present different rescheduling approaches on a numerical example. In Sect. 3, we give the description of the rescheduling environment and formulations for the considered rescheduling problems. In Sect. 4, we discuss alternative formulations for the problems by using conic quadratic inequalities. In Sect. 5, we propose a heuristic algorithm to generate approximately efficient solutions for the problems. We give the results of our computational study in Sect. 6 and finalize with concluding remarks in Sect. 7.

2 Rescheduling with controllable processing times: a numerical example

In order to clarify the distinctions between match-up rescheduling with fixed and controllable times, we demonstrate alternative rescheduling approaches on a numerical example in Fig. 1. This is a parallel machine rescheduling example arising due to a breakdown on one of the machines. In this study, the problem is to reschedule a set of jobs on non-identical machines where each job has different cost and processing time data on different machines. However, in the numerical example, for simplicity we assume that the jobs are of the same type and the machines are identical. Hence, we assume $p_{ij}^u = 2.0$ and $u_{ij} = 1$ for all i, j. The processing time of the jobs can be compressed by incurring additional manufacturing cost determined by the compression cost function which is $f_{ij} = 5y_{ij}^2$ for all i, j. The example starts with a preschedule with the minimum total compression cost. There are 15 jobs scheduled on three parallel CNC machines. When there is a machine breakdown on one of the machines, we illustrate how alternative rescheduling approaches can be used to remedy this unavailability period using a different Gantt Chart representation for each one of them.

Gantt Chart 1 in Fig. 1 belongs to the preschedule. In order to obtain this preschedule, we first solve a machine-job assignment problem with cost minimization objective. Thus, we find the optimum machine-job assignments (5 jobs on each machine) and compression levels (0.2) on the processing times subject to given capacities on the machines. Selected machine-job allocation and processing times have to satisfy the available machining time capacity constraint, which is taken as (9.0) for this numerical example.

Gantt Chart 2 shows the breakdown on machine 1. In practice, we do not know when a machine breakdown occurs, but we can determine when it ends right after its occurrence. Therefore, we assume that the breakdown time is not known a priori, but immediately after the event occurs the down duration can be determined. In this numerical example, a breakdown occurs at the start time of job 2. At this point, the jobs 1, 6 and 11 have been completed, and some are in process on other machines. These jobs, which are highlighted on the chart, will not be considered in rescheduling decisions since they are already completed. Gantt Chart 2 also gives the resulting schedule achieved by right-shifting the jobs not yet finished at time of breakdown on machine 1. The right-shift approach assumes fixed processing times and it is the simplest rescheduling strategy. However, the resulting schedule violates the machining time capacity constraint by the duration of down time, which is 3.60. The total manufacturing cost of the jobs considered in rescheduling stays the same (3.0) as their cost in the preschedule since the processing times did not change. Since there is no inserted idle time in the preschedule, we will not be able to find a feasible solution by looking for an alternative machine-job allocation with fixed processing times. The only way to find a feasible schedule without violating the machining time capacity constraints for this particular example is to consider the controllable processing times and machine reallocation decisions simultaneously.

Gantt Charts 3 to 6 in Fig. 1 next show alternative rescheduling approaches with controllable processing times. The first approach finds a schedule with minimum sum of match-up times on the machines. In other words, the objective is to minimize the length of the rescheduled portion of

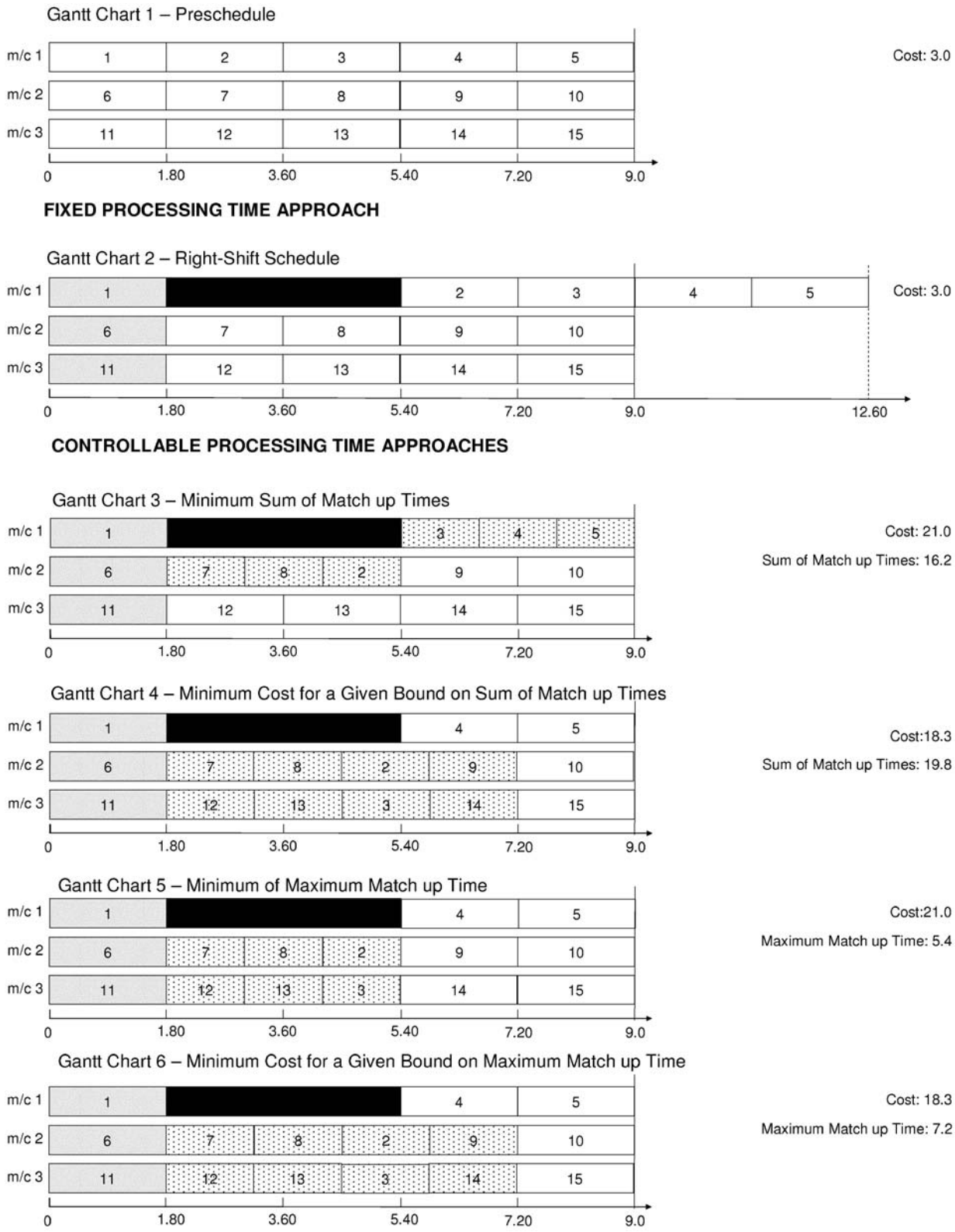

Fig. 1 Alternative reactive scheduling approaches

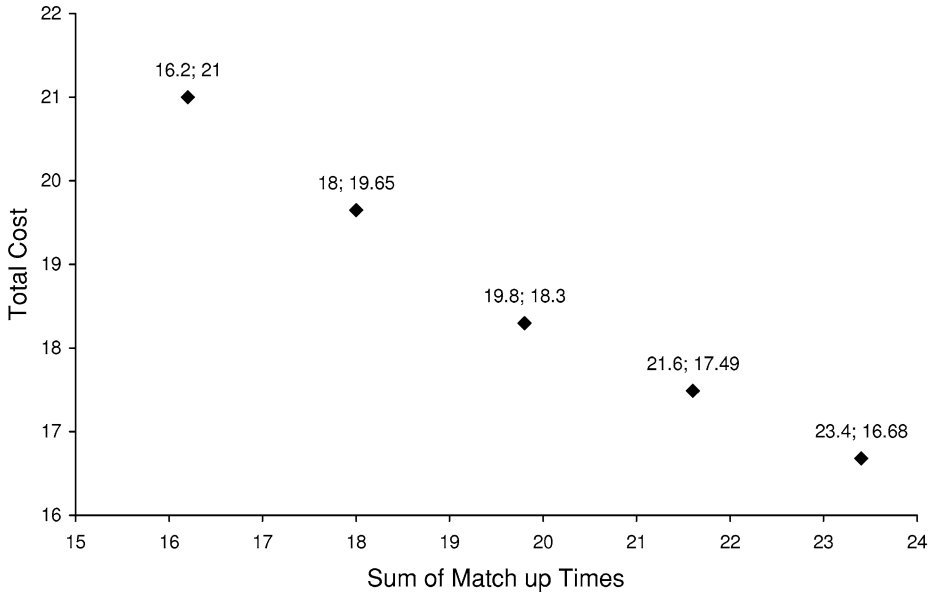

Fig. 2 Efficient solution set for total cost and sum of match-up times objectives

the preschedule. Minimum sum of match-up times for the schedule shown in Gantt Chart 3 is 16.20. This schedule is achieved by moving job 2 to machine 2 and re-planning the processing times of only six jobs represented by the dotted boxes. This is a minimum cost schedule for the sum of match-up time level of 16.20. As a result, the match-up times on machine 1, machine 2 and machine 3 are the end time of job 5, the starting time of job 9, and the starting time of job 12 in the preschedule, respectively. As it can be observed, the schedule on each machine is exactly the same as the preschedule beyond the match-up points. In contrast to the fixed processing time approach given in the previous charts, this approach neither violates the machining time capacity constraint nor leaves unnecessary idle time. This schedule fully utilizes the available machining time capacity on the machines and it has a manufacturing cost of 21.0.

Gantt Chart 4 in Fig. 1 presents the schedule achieved by minimizing the manufacturing cost for a given upper bound on the sum of match-up times. Sum of match-up times for the schedule is 19.80 and the total cost is 18.30. Hence, compared to the schedule in chart 3, by giving up from the sum of match-up times performance, which implies distributing the compression on more jobs, we improve the cost criterion. We re-assigned jobs 2 and 3 to different machines compared to the preschedule in this case. Gantt Chart 5 gives the schedule which minimizes the maximum of match-up times on the machines. Minimum of maximum match-up times is 5.40 for the given example. Minimizing the total cost subject to a maximum match-up time level of 7.20, we find the schedule shown in Gantt Chart 6 in Fig. 1.

Alternative rescheduling approaches applied in this example show that using processing time controllability has definite advantages. The first one is that we can catch up the preschedule soon after the breakdown occurs and satisfy the due dates and production plans in the system. Processing time and machine-job assignment changes affect only a small part of the preschedule which helps to decrease the stress in the system. Under the fixed processing times assumption, it may not be possible to catch up the preschedule as shown in the numerical example. Secondly, we have the flexibility to generate different alternative schedules with varying total cost and match-up time objectives. We can provide a set of alternative schedules to the decision maker. Thirdly, process time controllability provides a more robust system. With fixed processing times, a breakdown may result in a rescheduling problem which is infeasible even for a preschedule with idle times.

For the considered numerical example, we present a set of efficient solutions illustrating the tradeoff between the sum of match-up times and total cost objectives in Fig. 2. Each solution on the chart is labeled by its sum of match-up times value and total cost value, respectively. The first solution corresponds to the schedule in Chart 3 with the minimum sum of match-up times but a high total cost. The third solution corresponds to the schedule in Chart 4. Similarly, we present a set of efficient solutions for the minimum of maximum match-up times and the manufacturing cost objectives in Fig. 3. The first two solutions correspond to the schedules given in Charts 5 and 6, respectively. The third efficient solution shown in the figure has the minimum manufacturing cost for the problem. In the next section, we will describe the scheduling environment in detail and give problem definitions for different rescheduling objectives.

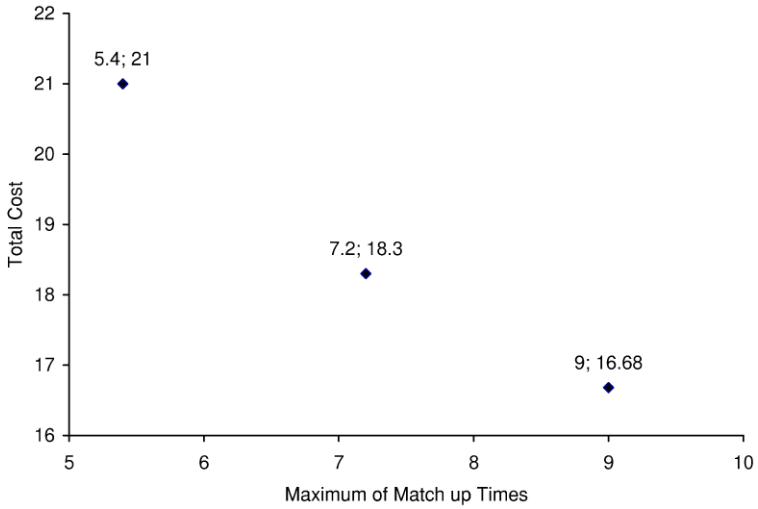

Fig. 3 Efficient solution set for total cost and minimum of maximum match-up time objectives

3 Scheduling environment and problem definitions

We consider n jobs to be processed in non-identical parallel CNC machine environment. The term "non-identical" means that each job may have different processing time, upper bound on compression and manufacturing cost values and different compression cost function on different machines. There are m machines, each of which has a given available machining time capacity. Due to different maintenance and allowance policies, each machine could have its own finite machining time capacity (in terms of fixed time units denoted as D_i) to process the set of assigned jobs. This allows us to have a more realistic representation of the rescheduling problems, since the production resources usually have limited capacities and scheduling problems should be solved over a given finite planning period. Furthermore, there is a given preschedule $\mathcal{S}$ being executed in this environment. As we have mentioned in Sect. 2, $\mathcal{S}$ could be formed by solving the machine-job assignment problem to minimize total manufacturing cost objective.

We assume that during the execution of $\mathcal{S}$, there occurs a breakdown on one of the machines. This means the machine will be down for a certain time to be maintained or repaired. Given such a disruption, $\mathcal{S}$ is no longer executable. We assume that if the machine fails in the middle of processing a job then this job has to be reprocessed in its entirety, called the preempt-repeat case in the literature. The interrupted job and all the other jobs which are not yet started processing on their machines have to be rescheduled. Rescheduling involves making new machine-job assignment and processing time compression decisions. These decisions can be made in order to catch up the preschedule at some point on each machine. Since we assume a non-preemptive machining environment, we select match-up times out of the start times previously determined in $\mathcal{S}$. The schedule, namely the sequence of the jobs and their start-end times, that follows a match-up time on a machine has to be the same as the preschedule. As simultaneous disruptions are infrequent compared to a single disruption, unless the system has serious maintenance issues, we do not consider multiple disruptions in this study.

The match-up time idea is used successfully to solve the rescheduling problems in the literature, but the critical issue is how to determine the match-up time on each machine. We could either minimize the sum of match-up times on each machine or minimize the maximum one regardless of their cost implications as will be discussed in Sects. 3.1 and 3.2. Another alternative could be to provide alternative match-up schedules with varying time/cost tradeoffs to the decision maker as will be discussed in Sects. 3.3 and 3.4. In the previous numerical example summarized in Fig. 1, by allowing machine reallocation and controllable processing times we could absorb the machine breakdown duration as soon as possible as in Gantt Charts 3 and 5 with two different match-up strategies albeit at a high manufacturing cost. With the maximum match-up time strategy, we expect to distribute the required compression amount more evenly among the selected jobs. This will eventually provide a better solution in terms of the manufacturing cost, but a higher number of jobs will be affected. We also suggest alternative nondominated solutions for this bi-criteria problem as given in Gantt Charts 4 and 6 for each strategy, respectively. In the following sections, we formulate alternative match-up scheduling strategies to deal with the stated time/cost tradeoff.

3.1 Minimize sum of match-up times

We first consider the problem of minimizing the sum of match-up times objective. This problem arises when the length of the rescheduled part of schedule $\mathcal{S}$ is critical and has to be minimized. Minimizing sum of match-up times can

be formulated as follows:

$$\min \quad \sum_i \sum_{j\in J_i^m} s_j z_j + \sum_i E_i \left(1 - \sum_{j\in J_i^m} z_j\right)$$

$$\text{s.t.} \quad \sum_{j\in J} \left(p_{ij}^u x_{ij} - y_{ij}\right) \le D_i, \quad i = 1,\ldots,m, \tag{1}$$

$$y_{ij} \le x_{ij} u_{ij}, \quad i = 1,\ldots,m \text{ and } j \in J, \tag{2}$$

$$\sum_{i=1}^{m} x_{ij} = 1, \quad \forall j \in J, \tag{3}$$

$$\text{(SM)} \quad \sum_{j\in J_i^m} z_j \le 1, \quad i = 1,\ldots,m, \tag{4}$$

$$\sum_{j_2\in P_{j_1}} z_{j_2} \le x_{ij_1}, \quad i = 1,\ldots,m \text{ and } \forall j_1 \in J_i^m, \tag{5}$$

$$\left(u_{ij_1} - y_{ij_1}^{\mathcal{S}}\right) \sum_{j_2\in P_{j_1}} z_{j_2} \le u_{ij_1} - y_{ij_1},$$
$$i = 1,\ldots,m \text{ and } \forall j_1 \in J_i^m, \tag{6}$$

$$y_{ij_1}^{\mathcal{S}} \sum_{j_2\in P_{j_1}} z_{j_2} \le y_{ij_1},$$
$$i = 1,\ldots,m \text{ and } \forall j_1 \in J_i^m, \tag{7}$$

$$x_{ij} \in \{0,1\}, \quad y_{ij} \in \mathbb{R}_+,$$
$$i = 1,\ldots,m, \text{ and } j \in J, \tag{8}$$

$$z_j \in \{0,1\}, \quad j \in \bigcup_i J_i^m. \tag{9}$$

The objective to minimize is the sum of match-up times. Possible match-up times are the start times of jobs which can still be started at the same time and on the same machine as in $\mathcal{S}$. For instance, in the numerical example, in Fig. 1, job 3 cannot form a match-up time since its machine is not available at that time, while job 10 can. End of horizon or ending time of the last job on a machine can also be the match-up time. Constraint set (1) guarantees that the new schedule does not exceed the available machining time capacity on each machine. Constraint set (2) is the variable upper bounding constraints on the amount of compression, guaranteeing that processing time of a job on a machine can be compressed only if the job is assigned on that machine and also the compression cannot be greater than the upper bound u_{ij}. Constraint set (3) assigns each job to a machine. Start time of at most one of the jobs can be selected as a match-up point on each machine which is forced by constraint set (4). Constraint set (5) guarantees that the jobs following a selected match-up time on a machine has to stay on the same machine in the new schedule. Constraints (6) and (7) fix the compression on a job which follows a match-up point at its compression in $\mathcal{S}$.

In Fig. 1, for the considered example, solution to the above problem is given in Gantt Chart 3. Selected match-up points on the machines are the end time of job 5 on machine 1 and the start times of jobs 9 and 12 on machines 2 and 3, respectively. The minimum sum of match-up times is the sum of these selected match-up points. As can be seen from the example, the new schedule following the match-up points is exactly the same as the preschedule. The length of the rescheduled part is at its minimum. Another critical performance criterion is the maximum of match-up times which is considered in the next section.

3.2 Minimize maximum of match-up times

It may also be critical for the decision maker to catch the preschedule on all machines as soon as possible. Then, his objective will be to minimize the latest match-up time on the machines. We can formulate this problem as follows:

$$\min \quad W$$

$$\text{(MM)} \quad \text{s.t.} \quad \sum_{j\in J_i^m} s_j z_j + E_i \left(1 - \sum_{j\in J_i^m} z_j\right) \le W,$$
$$i = 1,\ldots,m, \tag{10}$$
$$\text{and (1)–(9).}$$

Constraints (10) bound the match-up time on each machine from above by the variable W, which is minimized. The other constraints are the same as the sum of match-up times problem. Solving the rescheduling problem for either of the two match-up time related objectives that we have discussed so far results in extremely compressed processing times and costly machine-job assignments. Hence, the manufacturing cost for the resulting schedule could be too high. In the following sections, we will consider the objective of total manufacturing cost, while bounding the match-up time criterion of the schedule.

3.3 Minimize total manufacturing cost subject to a bound on sum of match-up times

Compressing the processing time of a job requires using additional resources for the job. In a flexible manufacturing system, reducing the processing time of an operation on a CNC machine by increasing the cutting speed and feed rate naturally leads to reduced tool life, and, consequently, increased machining cost. We can model the change in the machining cost due to processing time compression $y \ge 0$ as

$$f(y) = k y^{a/b},$$

where a and b are integers satisfying $a \ge b > 0$ and $k > 0$, so that f is an increasing and convex function of the compression. The function f reflects the relationship between

compression and cost in that as one decreases the processing time of a job, it becomes more expensive to compress it further. Convexity of f models the increasing marginal cost of compression. Technical specifications of a job such as its length, diameter, required surface quality, as well as machine and tool type determine the cost coefficients k, a, and b as discussed in Kayan and Aktürk (2005).

Minimizing the manufacturing cost necessitates making appropriate machine-job assignment and compression decisions. In match-up rescheduling, manufacturing cost due to process time compression and match-up time objectives are in conflict. Increasing the match-up time window on a machine not only allows one to distribute the required compression across a larger number of jobs, but also provides new machine-job reassignment possibilities. As this relaxes the feasible region of the problem it helps to reduce the manufacturing cost. Having two conflicting objectives, one way to find efficient solutions is to minimize one of the objectives subject to the constraint that the solution value of the second objective cannot be worse than the given upper bound, and solve the overall problem as a single objective problem. This method known as the ϵ-constraint approach, as discussed in T'kindt and Billaut (2006), has been widely used in the literature, because the decision maker can interactively specify and modify the bounds and analyze the influence of these modifications on the final solution. Below, we formulate the problem of minimizing the manufacturing cost subject to an upper bound T on the sum of match-up times in the new schedule:

$$\min \quad \sum_i \sum_{j\in J} \left(c_{ij}x_{ij} + f_{ij}(y_{ij})\right)$$

$$\text{(CSM)} \quad \text{s.t.} \quad \sum_i \sum_{j\in J_i^m} s_j z_j + \sum_i E_i \left(1 - \sum_{j\in J_i^m} z_j\right) \le T, \tag{11}$$

and (1)–(9).

The objective function above is the sum of fixed costs and compression costs. The formulation above includes convex functions in the objective. Constraint (11) sets the upper bound on the sum of match-up times for the schedule and, thus, limits the size of the rescheduled portion of the preschedule.

3.4 Minimize total manufacturing cost subject to a bound on maximum match-up time

Given an upper bound T on the maximum match-up time, one can set the match-up time on machine i to be $T_i = \max_{j\in J_i}\{s_j : s_j \le T\}$. This is due to the fact that increasing the match-up time on a machine does not increase the total manufacturing cost of a schedule as discussed above. Defining the set of jobs that precede selected match-up times on their machines as J^T, we can formulate the problem of minimizing manufacturing cost subject to a given maximum match-up time as:

$$\min \quad \sum_i \sum_{j\in J^T} \left(c_{ij}x_{ij} + f_{ij}(y_{ij})\right)$$

$$\text{s.t.} \quad \sum_{j\in J^T} \left(p_{ij}^u x_{ij} - y_{ij}\right) \le T_i, \quad i = 1,\ldots,m, \tag{12}$$

$$\text{(CMM)} \quad y_{ij} \le x_{ij}u_{ij}, \quad i = 1,\ldots,m \text{ and } j \in J^T, \tag{13}$$

$$\sum_{i=1}^{m} x_{ij} = 1, \quad j \in J^T, \tag{14}$$

$$x_{ij} \in \{0,1\}, \quad y_{ij} \in \mathbb{R}_+, \quad i = 1,\ldots,m \text{ and } j \in J^T. \tag{15}$$

In this section, we have described the rescheduling environment and provided four different rescheduling problems to solve. In the computational analysis section, we will demonstrate that minimizing the sum of match-up times and minimizing maximum match-up time problems can be solved easily by commercial mixed-integer programming (MIP) solvers. However, it is usually difficult to solve cost minimization problems since they have convex cost terms in their objective functions. In the next section, we will discuss how the cost minimization problems can be reformulated in a stronger way by using conic quadratic inequalities.

4 Strong conic quadratic formulations for cost minimization problems

When solving the continuous relaxation of CSM and CMM formulations, convex cost terms in the objectives cause highly fractional optimal solutions which lie in the interior of their convex relaxation. Hence, branch-and-bound algorithms based on such relaxations usually require excessive branching to find integer feasible solutions. However, in reactive rescheduling, solution times for the problems are quite critical. Recent work by Aktürk et al. (2009) on problems with a separable convex objective and variable upper bounding constraints shows that conic quadratic inequalities can be employed for strengthening the formulations of such problems. Their approach is based on second-order cone programming (SOCP), which is also called conic quadratic programming. Alizadeh and Goldfarb (2003) give an extensive review on the theory of SOCP and report the recent advances in this area. Existence of efficient SOCP algorithms implemented in branch-and-bound solvers allows

us to reformulate CSM and CMM models as proposed by Aktürk et al. (2009) and solve them efficiently. In this section, we discuss the implementation of this approach in the rescheduling problems with cost minimization objectives.

The first step of the reformulation is to put CSM into conic optimization problem form with linear objective and conic constraints. Thus, we replace each term $y_{ij}^{a_{ij}/b_{ij}}$ in the objective with an auxiliary variable t_{ij} and add $y_{ij}^{a_{ij}/b_{ij}} \le t_{ij}$ into the constraints as below:

$$\begin{aligned} &\min \quad \sum_i \sum_{j\in J} (c_{ij}x_{ij} + k_{ij}t_{ij}) \\ \text{(CSM1)} \quad &\text{s.t.} \quad y_{ij}^{a_{ij}/b_{ij}} \le t_{ij}, \quad i = 1,\dots,m,\ j \in J, \quad (16) \\ &\qquad \text{and (1)–(9) and (11).} \end{aligned}$$

As $c_{ij}, t_{ij} \ge 0$ and $b_{ij} > 0$ for all i, j, Inequality (16) is equivalent to

$$y_{ij}^{a_{ij}} \le t_{ij}^{b_{ij}}, \quad (17)$$

which can be strengthened as

$$y_{ij}^{a_{ij}} \le t_{ij}^{b_{ij}} x_{ij}^{a_{ij}-b_{ij}}. \quad (18)$$

Because $a_{ij} \ge b_{ij}$, for $0 \le x_{ij} \le 1$, Inequality (18) implies (17). Thus, we can substitute the inequalities with the stronger ones and obtain:

$$\begin{aligned} &\min \quad \sum_i \sum_{j\in J} (c_{ij}x_{ij} + k_{ij}t_{ij}) \\ \text{(CSM2)} \quad &\text{s.t.} \quad y_{ij}^{a_{ij}} \le t_{ij}^{b_{ij}} x_{ij}^{a_{ij}-b_{ij}}, \\ &\qquad i = 1,\dots,m,\ j \in J, \quad (19) \\ &\qquad \text{and (1)–(9) and (11).} \end{aligned}$$

Inequalities (17) and (18) can be represented by using conic quadratic constraints. This can be shown first by using the fact that an inequality of the form

$$r^{2^l} \le s_1 s_2 \cdots s_{2^l}, \quad (20)$$

for $r, s_1, \dots, s_{2^l} \ge 0$, can be expressed equivalently using $O(2^l)$ variables and $O(2^l)$ hyperbolic inequalities of the form

$$u^2 \le v_1 v_2, \quad u, v_1, v_2 \ge 0 \quad (21)$$

(Ben-Tal and Nemirovski 2001), and then using the fact that each constraint (21) can be written as a second-order conic constraint

$$\|(2u, v_1 - v_2)\| \le v_1 + v_2, \quad (22)$$

where $\|v\|$ denotes the Euclidean norm of vector v. Aktürk et al. (2009) have shown that Inequalities (17) and (18) can be represented equivalently by using $O(\log_2 a_{ij})$ variables and $O(\log_2 a_{ij})$ conic quadratic constraints of the form (22). An example for the conic quadratic representation of a given Inequality (18) is shown below.

Example 1 Consider the convex function $f(y) = y^{5/4}$. We first write inequality (18) as

$$y^5 \le t^4 x, \quad y, t, x \ge 0$$

then put it in the form of (20) as

$$y^8 \le t^4 x y^3, \quad y, t, x \ge 0,$$

which we can express equivalently by using the following hyperbolic constraints:

$$\begin{aligned} &v_1^2 \le xy, \quad x, y \ge 0, \\ &v_2^2 \le y v_1, \quad y, v_1 \ge 0, \\ &y^2 \le t v_2, \quad t, v_2 \ge 0. \end{aligned}$$

The hyperbolic constraints are then written in conic quadratic form as

$$\begin{aligned} &\|(2v_1, y - x)\| \le y + x, \\ &\|(2v_2, y - v_1)\| \le y + v_1, \\ &\|(2y, t - v_2)\| \le t + v_2. \end{aligned}$$

We have discussed the strengthening and reformulation on CSM model. CMM can also be reformulated in the same way. Using strengthened conic quadratic formulations allows one to solve CSM and CMM very quickly as discussed in Sect. 6. In the next section, we describe a heuristic approach to generate approximately efficient solutions for the cost and the match-up objectives.

5 Generating a set of approximately efficient solutions: heuristic approach

In the rescheduling literature, a common approach to solve scheduling problems is by heuristics which can find a solution quickly. In the previous section, we have discussed a way of getting strong formulations for the considered cost minimization problems, CSM and CMM. In this section, we propose a heuristic search algorithm which generates a set of approximately efficient solutions that can be presented to the decision maker quickly. We consider two bi-criteria rescheduling problems. In the first problem, the conflicting objectives are the total manufacturing cost and the sum of match-up times; whereas in the second one, the conflicting

Algorithm 1 Heuristic algorithm for finding approximate efficient solutions

Require: A preschedule $\mathcal{S}$ and a disruption on one of the machines.
Solve the problem SM (MM) to find an initial schedule S, match-up times T_i, and job pool P.
repeat
Apply `1-move` Algorithm.
Apply `2-swap` Algorithm.
Report the generated solution.
until `Augment Job Pool` is `False`.

objectives are the total manufacturing cost and maximum match-up time.

We first give a step by step definition of the proposed heuristic algorithm below. The algorithm generates approximately efficient solutions for the bi-criteria match-up scheduling problems under consideration.

Algorithm 1 starts with an initial schedule $\mathcal{S}$ and a given disruption. We first solve the problem of minimizing sum of match-up times (or minimizing maximum of match-up times). The solution for this problem gives the initial match-up times and a job pool, i.e., the set of incomplete jobs that precede the match-up times. With the given match-up times, the algorithm applies the `1-move` and `2-swap` improvement algorithms on the current job pool and records the solution. The next step is to expand the job pool by adding an appropriate job to the pool and the improvement algorithms are applied on the new job pool. The algorithm terminates when the job pool cannot be expanded, i.e., no jobs exist to be added to the pool. Algorithm 1 generates a set of solutions where each solution is an approximately efficient solution. In the following, we will describe a subproblem which motivates the improvement search steps of the algorithm and then we will describe the steps of Algorithm 1 in detail.

5.1 A subproblem

We first define a subproblem that is solved at each iteration of Algorithm 1. The solution to the subproblem is used by the `1-move` and `2-swap` improvements and for augmenting the job pool. The subproblem for machine i is described as the following: Given a match-up time T_i and a set of jobs J_c to be scheduled before T_i on machine i, find optimal compression levels for the jobs so that the total compression cost is minimized. The problem is formulated as

$$\text{(CIMPi)} \quad \min \sum_{j\in J_c} f_{ij}(y_{ij})$$

$$\text{s.t.} \sum_{j\in J_c} \left(p_{ij}^u - y_{ij}\right) \le T_i, \tag{23}$$

$$0 \le y_{ij} \le u_{ij}, \quad j \in J_c. \tag{24}$$

COMPi is a nonlinear continuous resource allocation problem. Optimality properties and a solution method for the problem were given by Bretthauer and Shetty (1995). We can write the Karush–Kuhn–Tucker conditions for COMPi as below:

$$\frac{\partial f_{ij}}{\partial y_{ij}} - \lambda - \nu_j + \eta_j = 0, \quad j \in J_c, \tag{25}$$

$$\lambda\left(\sum_{j\in J_c}\left(p_{ij}^u - y_{ij}\right) - T_i\right) = 0, \tag{26}$$

$$\nu_j y_{ij} = 0, \quad j \in J_c, \tag{27}$$

$$\eta_j(y_{ij} - u_{ij}) = 0, \quad j \in J_c, \tag{28}$$

$$\nu_j \ge 0, \quad \eta_j \ge 0, \quad j \in J_c, \tag{29}$$

$$\lambda \ge 0, \tag{30}$$

and Inequalities (23)–(24). These conditions imply the following lemma which will be very useful in designing improvement steps and augmenting the job pool in our heuristic.

Lemma 1 *Let y_{ij}^* and $(\lambda^*, \eta^*, \nu^*)$ be an optimal pair of solutions to COMPi. For each job j, we have the following*:

$$(\partial f_{ij}/\partial y_{ij})\left(y_{ij}^*\right) \begin{cases} \ge \lambda^*, & \text{if } y_{ij}^* = 0; \\ = \lambda^*, & \text{if } 0 < y_{ij}^* < u_{ij}; \\ \le \lambda^*, & \text{if } y_{ij}^* = u_{ij}. \end{cases}$$

Because $\lambda^* \ge 0$ and $(\partial f_{ij}/\partial y_{ij})(0) = 0$, the first part holds only when $\lambda^* = 0$. Thus Lemma 1 states that, whenever $\lambda^* > 0$, the partial derivative of the cost function for each job must be equal unless its compression is at its upper bound u_{ij}. Here, λ^* is the rate of change of the optimal cost as T_i changes. But also from the lemma, the rate of change of the optimal cost for each job with $0 < y_{ij}^* < u_{ij}$ is also equal to λ^*. This interpretation will be used in designing search steps in the following sections.

5.2 Job pool

In the heuristic algorithm, at each iteration we work on a job pool which is the set of jobs to be rescheduled at that iteration. Given match-up times on the machines, a job is included in the job pool if it is not yet started at the time of the

breakdown and its start time precedes the given match-up time on its machine. The only exception is the interrupted job on the breakdown machine which is included in the job pool. For the bi-criteria problem with cost and sum of match-up times objectives, the initial job pool is determined by the match-up points found by solving the SM problem in Sect. 3.1. Solving SM gives us the minimum attainable sum of match-up times. As shown in Gantt Charts 3 and 5 in Fig. 1, minimizing sum of match-up times or maximum of match-up times requires extensively compressing a small set of jobs. The solution with minimum sum of match-up times has the highest manufacturing cost. Thus, the first approximately efficient solution we generate has the minimum possible sum of match-up times but its manufacturing cost is the worst of all efficient solutions.

For the problem with the objectives of minimizing total cost and maximum match-up time, we get the initial job pool by solving the MM problem and setting the match-up time on each machine to be the highest match-up time less than the maximum match-up time found by solving MM.

We augment the job pool at each iteration by adding a new job to the pool. The question is which job to add to the current job pool. Considering the jobs which are immediate successors of match-up points; there are at most m candidates. Adding a new job to the job pool increases the sum of match-up times, but it may give us a schedule with a lower manufacturing cost after reallocating the jobs and solving the subproblems on each machine. We use the following rule to select the machine for increasing its match-up time. For each machine i, we have λ_i^*, the rate of change of the optimal cost by the change of match-up time, for the jobs scheduled before match-up point. Suppose that the job that immediately succeeds match-up point on machine i is j and that the compression on job j is y_{ij}^*, then we select the machine with the smallest

$$\Delta_i := \frac{k_{ij}\hat{y}_{ij}^{a_{ij}/b_{ij}} - k_{ij}y_{ij}^{*\,a_{ij}/b_{ij}} - \lambda_i^*(\hat{y}_{ij} - y_{ij}^*)}{p_{ij}^u - y_{ij}^*},$$

where $\hat{y}_{ij} = \min((\partial f_{ij}/\partial y_{ij})^{-1}(\lambda_i^*), u_{ij})$. Δ_i is an estimate of the ratio of the cost change to the match-up time change that will be obtained by moving the match-up point to the start time of the next job. Consider machine 2 in the schedule represented by Gantt Chart 4 in Fig. 1. On machine 2, $\lambda_2^* = 6.5$, $y_{2,10}^* = 0.2$ and $\hat{y}_{2,10} = 0.65$. Then, $\Delta_2 = \frac{5\times(0.65)^2 - 5\times(0.2)^2 - 6.5\times(0.65-0.2)}{1.8} = -2.25$. The first two terms in the numerator give the cost change that would occur if job 10 were compressed by the same amount as the other jobs in the job pool on machine 2. The third term gives an estimate for the cost change by expanding the jobs in the job pool on machine 2 by the difference between two compression levels for job 10 considered in the first two terms. In other words, we first increase the compression of job 10 by 0.45 and then we expand jobs 2, 7, 8, 9, and 10 by a total amount of 0.45. Hence, Δ_2 estimates the cost improvement that would occur by adding job 10 to the job pool and then replanning the compression levels of the jobs. Since we are interested in the efficient frontier of manufacturing cost and sum of match-up time objectives, we select the machine that gives the maximum cost improvement estimate per unit match-up time change as outlined in Algorithm 2.

Algorithm 2 Augment job pool

Require: Given match-up times T_i for each machine and job pool P.
if $T_i = E_i$ for all i **then**
  return `False`.
else
  Calculate Δ_i for the machines with $T_i \neq E_i$.
  Select i^* with minimum Δ_i.
  $T_{i^*} \leftarrow s_j$, where j is the next job on i^*; $P \leftarrow P \cup \{j\}$.
  return `True`.

For the maximum match-up time criterion, the match-up time increasing rule we use is to select the candidate job which has the smallest completion time. We have discussed the compression subproblem, formation of initial job pool and job pool augmentation rules used in our heuristic. We next describe the improvement search methods.

5.3 1-move improvement search

`1-move` method assumes that we have a schedule at hand with given match-up times and that the compression subproblem COMPi is solved for each machine. `1-move` method looks for cost improving move of a job in the job pool from its current machine to another machine. A `1-move` yields compression cost improvement in its original machine since the compression for the remaining jobs can be decreased due to the additional machining time capacity that becomes available when the job leaves. On the other hand, it increases the compression cost on the new machine as the jobs on this machine need to be compressed further to make up space for the new job to get a feasible schedule. Below we give a lower bound on the cost change for a given `1-move`.

Lemma 2 (Lower bound for a `1-move`) *For a given schedule let λ_{i_1} and λ_{i_2} be optimal dual prices for COMPi$_1$ and COMPi$_2$, respectively, and $y_{i_1 j}$ be the compression of job j on machine i_1. Then, a lower bound for the cost change that will result by moving job j from machine i_1 to i_2 is as stated below*:

$$LB\big(j : (i_1 \to i_2)\big) = -\lambda_{i_1}(p_{i_1 j} - y_{i_1 j}) - c_{i_1 j} - f_{i_1 j}(y_{i_1 j}) + c_{i_2 j} + f_{i_2 j}(\hat{y}_{i_2 j}) + \lambda_{i_2}(p_{i_2 j} - \hat{y}_{i_2 j}),$$

Algorithm 3 `1-move` search algorithm

Require: A given schedule S and a job pool P.
  repeat
    Generate all feasible `1-moves` for each j in P in S;
    Calculate LB for all moves;
    if $LB \geq 0$ for all feasible moves **then**
      BREAK;
    else
      Make a list of moves with $LB < 0$ in nondecreasing order of LBs.
      Initialize: `found_improving_move` $\leftarrow$ `False`, `end_of_list` $\leftarrow$ `False`;
      while NOT `found_improving_move` and NOT `end_of_list` **do**
        Do the next move in the list;
        if It is the last move in the list **then**
          `end_of_list` $\leftarrow$ `True`
        Solve COMPi for affected machines;
        New schedule is S';
        if $COST(S') < COST(S)$ **then**
          $S \leftarrow S'$;
          `found_improving_move` $\leftarrow$ `True`;
          `improved` $\leftarrow$ `True`;
      if NOT `found_improving_move` **then**
        `improved` $\leftarrow$ `False`
  until NOT `improved`.

where $\hat{y}_{i_2 j} = \min((\partial f_{i_2 j}/\partial y_{i_2 j})^{-1}(\lambda_{i_2}), u_{i_2 j})$.

Proof The first three terms in $LB(j : (i_1 \rightarrow i_2))$ give a lower bound on the cost reduction by removing job j from machine i_1; whereas the last three terms give a lower bound on the cost increase by inserting job j into machine i_2. □

$LB(j : (i_1 \rightarrow i_2))$ gives us a lower bound on the change of manufacturing cost change for moving job j from i_1 to i_2. So if the lower bound is positive, then the corresponding `1-move` does not improve the cost of the schedule. On the other hand, if it is negative, then it may be possible to improve the cost of the schedule by reallocating this job to machine i_2. Hence, we employ a procedure to implement one moves on a given schedule as given in Algorithm 3.

Algorithm 3 starts with an initial schedule and applies promising `1-moves` iteratively to obtain schedules with improved total manufacturing cost. The algorithm terminates when no improvement is possible for the current schedule. In the next section, we give a larger neighborhood search by a `2-swap` move.

5.4 2-swap improvement search

A `2-swap` move is the exchange of two jobs, j_1 and j_2, between two machines i_1 and i_2, i.e., moving job j_1 from machine i_1 to i_2, and job j_2 in the opposite direction. We can consider a `2-swap` move as a combination of two `1-move`'s and calculate a cost change lower bound for a given 2-move as below:

Lemma 3 (Lower bound for a `2-swap`) *For a given schedule let λ_{i_1} and λ_{i_2} be optimal dual prices for COMPi$_1$ and COMPi$_2$, respectively, and $y_{i_1 j_1}$ and $y_{i_2 j_2}$ be the compression of the jobs j_1 and j_2 on machine i_1 and i_2, respectively. Then, a lower bound for the cost change that will result by swapping jobs j_1 and j_2 between machines i_1 and i_2 is calculated as below*:

$$\begin{aligned} LB(j_1 \leftrightarrow j_2) = {} & \lambda_{i_1}(p_{i_1 j_1} - y_{i_1 j_1} - p_{i_1 j_2} + \hat{y}_{i_1 j_2}) - c_{i_1 j_1} \\ & - f_{i_1 j_1}(y_{i_1 j_1}) + c_{i_1 j_2} + f_{i_1 j_2}(\hat{y}_{i_1 j_2}) \\ & + \lambda_{i_2}(p_{i_2 j_2} - y_{i_2 j_2} - p_{i_2 j_1} + \hat{y}_{i_2 j_1}) - c_{i_2 j_2} \\ & - f_{i_2 j_2}(y_{i_2 j_2}) + c_{i_2 j_1} + f_{i_2 j_1}(\hat{y}_{i_2 j_1}), \end{aligned}$$

where $\hat{y}_{i_2 j_1} = \min((\frac{\partial f_{i_2 j_1}}{\partial p_{i_2 j_1}})^{-1}(\lambda_{i_2}), u_{i_2 j_1})$ *and* $\hat{y}_{i_1 j_2} = \min((\frac{\partial f_{i_1 j_2}}{\partial y_{i_1 j_2}})^{-1}(\lambda_{i_1}), u_{i_1 j_2})$.

Proof Similar to the proof of Lemma 2. □

As in the `1-move` case, if the lower bound for a given `2-swap` is positive, then the `2-swap` does not reduce the cost of the current schedule. However, a `2-swap` with a negative lower bound has the potential for improvement. So starting from the most promising `2-swap`, we try possible two swaps for a given schedule and improve it iteratively. Hence, the algorithm for `2-swap` improvement search is same as Algorithm 3 except that `2-swaps` are considered instead of `1-moves`.

In this section, we have described a heuristic algorithm which generates a set of approximately efficient solutions for each bi-criteria problem considered. Each iteration of the algorithm gives a new solution with increased match-up times, and for each new solution the manufacturing cost is either decreased or stays the same. At the end, Algorithm 1 produces a set of solutions which approximate the efficient frontier of match-up time versus manufacturing cost trade-off. In the next section, we will describe our computational results on this heuristic method and the exact solution approaches.

6 Computational study

In the computational study, we tested the performance of alternative conic quadratic formulations introduced in Sects. 3 and 4 for generating exact efficient solutions and

of the heuristic algorithm described in Sect. 5 for generating approximate efficient solutions. We compared the computation time and solution quality of these alternative approaches on a set of randomly generated test problems. The test problems have varying number of jobs ($n = 50, 100$), machines ($m = 2, 3$), capacity factor ($\kappa = 0.25, 0.30$), and average length of disruption ($ld = 2.0, 5.0$). For each of the 16 possible configurations of these four parameters, we generated 15 problem instances at random. The fixed cost (c_{ij}) for each job-machine pair was generated from Uniform [2.0, 6.0]. The coefficients of the compression cost function ($f_{ij}(y_{ij}) = k_{ij} y_{ij}^{a_{ij}/b_{ij}}$) k_{ij} were generated from Uniform [1.0, 3.0] and $a_{ij} = U/10$, where U is from a discrete Uniform [11, 31] and $b_{ij} = 1$. Processing time p_{ij}^u was generated from Uniform [1.0, 3.0], whereas $u_{ij} = p_{ij}^u \times U$, where U is from Uniform [0.5, 0.9]. We set the machining time capacity of each machine equal to

$$D_i = \kappa \times \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij}^u}{m}.$$

In order to construct preschedules, we first solved the machine-job assignment problem that minimizes total manufacturing cost subject to given machining time capacity for each machine. We sequenced the allocated jobs on each machine by using the shortest processing time (SPT) first rule, which gives the minimum total completion time for a given set of jobs on a machine. In the preschedules generated for the experiments, the ratio of the amount of compression on a job to its maximum possible compression (y_{ij}^S / u_{ij}) is 38% on average, 68% at maximum and 6% at minimum. Hence, we tested our solution approaches on a wide range of preschedules with different compression levels.

A long disruption to a schedule may result in an infeasible rescheduling problem as it may not be possible to catch up a preschedule within given machining time capacity levels. Therefore, we limit our test problems to the ones that could lead to feasible rescheduling instances so that we could measure the objective function values along with the required computation times. On the other hand, it might be possible to have preschedules where machines have enough idle times to absorb the disruption durations. These schedules could be repaired without incurring any additional compression costs. In our computational study, we solve instances with no idle times, and hence our instances require additional compressions on processing times of jobs.

Having formed a preschedule, we generated a breakdown on the schedule by randomly selecting a machine and a job on the selected machine so that the breakdown occurs during the execution of the selected job and lasts for a duration generated from Uniform $[ld - 1.0, ld + 1.0]$. The rescheduling problem included the interrupted job on the breakdown machine and all jobs planned to be started processing at the time of breakdown or later in the preschedule.

For each problem instance, we first ran Algorithm 1 which generates a set of approximately efficient solutions. In order to test the solution quality of the points generated by Algorithm 1 against efficient solutions to be generated by CSM and CMM problems, we first selected three solutions out of the solution set generated by the algorithm. Let $\bar{T}_{\min}$ and $\bar{T}_{\max}$ be the minimum and maximum values of sum of match-up times achieved by the algorithm. We then calculate three different values of sum of match-up times by setting $k = 1, 2, 3$ in $\bar{T}_{\min} + k \times \frac{\bar{T}_{\max} - \bar{T}_{\min}}{4}$. For each value of k, we select a solution which has the closest possible sum of match-up times ($\bar{T}_k$) to the value calculated in the previous step. Then, for each k we solved CSM problem by setting $\bar{T} = \bar{T}_k$ and measured the relative gap between the cost of the heuristic solution and the cost of the exact efficient solution. We solved CSM by using the conic quadratic formulations, CSM1 and CSM2, given in Sect. 4. We followed the same approach for the maximum match-up time problem and solved formulations CMM1 and CMM2. All experiments were performed using ILOG CPLEX Version 10.1 on a 3 GHz Linux workstation with 512 MB memory with a 1000 CPU seconds time limit.

In the experimental runs, solving the machine-job allocation problem to form the preschedule took 16.14 CPU seconds on the average. After generating the disruptions, the average number of jobs to be rescheduled was 40.7 for the 50-job problems and 84.9 for the 100-job problems. Solving the SM model required 0.39 CPU seconds on average; the corresponding value for the MM model was 0.16. Thus, we can solve the sum of match-up times or maximum match-up time problems in very short CPU times.

In Table 1, we give the computational results for the conic quadratic representations of formulations CSM2 and CMM2. We report the number of problems (out of 15) solved to optimality within the time limit (opt). We also report the average relative gap between the best known upper bound and the lower bound at time of termination (egap), the average number of branch-and-bound nodes explored (node) and the average CPU time in seconds to solve the problems (cpu). We observe that CSM2 was able to solve 92% of the problems to optimality. CSM2 achieved a quite low optimality gap which is 0.07% in the worst case. Similarly, CMM2 could solve all the problems to optimality within short CPU times by exploring a small number of branch-and-bound nodes. We can even use CMM2 to generate all the solutions in the efficient frontier. Our computational results indicate that advances in conic mixed-integer programming provide us strong formulations which can solve rescheduling problems with controllable processing times within reasonable CPU times.

If the number of jobs or number of machines increases then the sizes of rescheduling problems increase as expected. Moreover, it takes more CPU time to solve the

Table 1 Results for strong conic quadratic formulations

ld	κ	*n*	*m*	Sum of match-up times CSM2				Maximum match-up time CMM2			
				opt	egap (%)	node	cpu	opt	egap (%)	node	cpu
2.0	0.25	50	2	15	0	35.7	9.8	15	0	2.9	1.3
			3	15	0	188.7	71.1	15	0	14.6	4.1
		100	2	15	0.01	106	55	15	0	0	0.6
			3	14	0.01	311.3	340.2	15	0	6.9	6.4
	0.30	50	2	15	0.01	41.3	12.6	15	0	2.6	1
			3	13	0.07	280.9	176.1	15	0	15.9	4.5
		100	2	15	0.01	82.5	58.1	15	0	2.4	2.1
			3	13	0.01	463.4	502.9	15	0	14	10.6
5.0	0.25	50	2	15	0	54.2	13.9	15	0	1.1	0.7
			3	14	0.03	306.9	161.8	15	0	10.2	3.5
		100	2	15	0.01	120.1	80	15	0	2.2	2
			3	11	0.05	624.4	567.6	15	0.01	11.7	7.9
	0.30	50	2	15	0	35.9	9.6	15	0	2.8	1
			3	14	0.01	496.1	213.7	15	0	25.1	7.7
		100	2	14	0.02	213.8	157.1	15	0	0.9	1.3
			3	8	0.05	710.2	792.7	15	0.01	17.6	14.2
Average				13.8	0.02	254.5	201.4	15	0.001	8.2	4.3

CSM2 formulation compared to the CMM2 formulation as shown in Table 1. Solving a CMM problem involves machine-job allocation decisions within given match-up times on the machines whereas solving a CSM problem involves making additional match-up time decisions as well. We can expect that solving CMM problems are easier than solving CSM problems. Table 1 shows that CMM problems were solved in 4.3 CPU seconds on average, whereas CSM problems took 201.4 CPU seconds to solve. We can also observe from Table 1 that when the length of disruption is higher it takes more time to solve cost minimization problems as sizes of the problems increase. Our computational results indicate that this effect is more notable in CSM problems.

In Table 2, we present the computational results for the heuristic algorithm, which generates approximate efficient solutions with varying cost and match-up time. In this table, we report the average number of solutions generated by the algorithm (# sol), and the average CPU time in seconds (cpu). Since we solved the CSM (CMM) problems for the sum of match-up times (maximum of match-up times) of the selected solutions generated by the heuristic, we can compare the manufacturing cost of the selected heuristic solutions with the corresponding optimal cost achieved by solving CSM (CMM). The relative gap is measured by $100 \times (z_H - z_{CSM})/z_{CSM}$, where z_H and z_{CSM} are the cost value of the heuristic solution and the optimal cost achieved by solving CSM, respectively. In the table, we report the mean, minimum and the maximum of the relative gap (gap).

The results show that Algorithm 1 runs within few seconds and generates a large number of alternative solutions with varying time/cost tradeoff to the decision maker. Such a solution set can be used to visualize an approximate efficient frontier. When we check the solution quality for cost versus sum of match-up times problem, we see that the average gap between the heuristic solution and the exact solution is less than 1% in most of the cases and is 1.58% at most. The worst gap performance is 6.23%, but in most of the cases it is less than 1.0%, implying an almost equivalent solution quality with exact approaches. For the maximum match-up time problem, the average gap for the heuristic is 0.73%. While the minimum gap can be as low as 0.0%, and we see the worst gap is 24.04%. For the cases where the decision maker may want to see the behavior of the efficient frontier quickly, the heuristic algorithm may be very valuable.

Table 2 Heuristic algorithm performance

ld	κ	*n*	*m*	Sum of match-up times					Maximum match-up time				
				# sol	cpu	gap (%)			# sol	cpu	gap (%)		
				mean	mean	mean	min	max	mean	mean	mean	min	max
2.0	0.25	50	2	19.8	1.26	0.04	0.00	0.21	27.6	2.05	0.28	0.00	1.06
			3	13.8	0.37	1.21	0.00	6.23	31.8	1.38	0.35	0.00	1.43
		100	2	70.2	8.58	0.21	0.00	0.66	67.6	12.03	0.17	0.08	0.24
			3	49	3.77	0.09	0.00	0.18	78.8	12.51	0.11	0.00	0.37
	0.30	50	2	18.6	0.64	0.39	0.00	1.92	31.4	2.30	0.21	0.00	0.62
			3	14.4	1.15	0.82	0.00	3.04	34.4	4.63	0.73	0.10	2.08
		100	2	50.4	3.77	0.09	0.00	0.55	72.6	10.22	1.17	0.00	15.75
			3	27.6	1.35	0.03	0.00	0.09	80.6	27.16	0.08	0.00	0.23
5.0	0.25	50	2	19.4	1.94	0.14	0.00	0.48	20.6	2.00	0.48	0.00	1.60
			3	21.2	1.16	0.49	0.00	1.41	25.0	1.72	1.94	0.00	18.00
		100	2	53.6	11.74	0.30	0.00	0.93	57.0	9.97	0.41	0.16	0.86
			3	57	10.33	0.26	0.00	0.92	70.2	15.60	0.38	0.07	1.04
	0.30	50	2	66.2	2.10	0.21	0.00	0.61	24.6	2.28	1.89	0.00	17.94
			3	19.4	1.23	1.58	0.17	4.11	31.2	2.44	0.36	0.00	1.25
		100	2	69.4	12.89	0.17	0.01	0.50	64.8	9.05	0.46	0.25	0.73
			3	48	6.26	0.97	0.00	2.68	74.0	24.80	2.64	0.09	24.04
Average				38.6	4.28	0.44	0.01	1.53	49.5	8.76	0.73	0.05	5.45

The results in Table 2 also indicate that the proposed heuristic algorithm performs well for different parameter settings. We can observe from the table that for both problems the average gap value tends to increase as the average disruption length is increased. As expected, the algorithm requires more CPU time as the number of jobs increases. As the length of disruption is increased the algorithm tends to generate more solutions for the sum of match-up times problem, since the algorithm adds more jobs to the job pool. When we check the maximum match-up time results, the number of alternative solutions decreases as the length of disruption is increased. This is due to the decreased number of possible match-up times in this case.

Finally, we have checked the effect of solving the sum of match-up time problem on the maximum match-up time objective and vice versa. If we solve the sum of match-up time problem, and check the maximum match-up time of the achieved schedules, we see that the resulting maximum match-up times deviate by 14.7% from optimum. Similarly, if we solve the maximum match-up time problem and check the sum of match-up times, we see a deviation of 25.5% from the optimum. We have also compared the manufacturing cost of the initial schedules achieved by the heuristic algorithm. The cost of the schedule achieved by solving maximum match-up time is 0.4% higher on average than the schedule achieved by solving sum of match-up times objectives. However, in terms of manufacturing cost, we do not see a statistically significant relationship between two approaches.

In this section, we have shown that we can efficiently solve match-up rescheduling problems with controllable processing times exactly by using recently developed reformulation techniques and commercial solvers. We have also observed that our heuristic is able to generate a very good approximation of the efficient frontier of match-up time and manufacturing cost quickly. In the next section, we conclude with some final remarks.

7 Conclusions

The results in this paper clearly indicate that controllable processing times introduce an important flexibility to rescheduling problems under a single machine breakdown. As a result of this solution flexibility, we can either generate schedules which can catch up the preschedule very

quickly after the disruption albeit at a high manufacturing cost, or we can distribute the effects of disruption to the entire schedule by creating alternative time/cost tradeoffs to the decision maker. We have introduced new match-up time related objectives, sum of match-up times and maximum match-up time, each which has its own advantages. It may be critical for the scheduler to balance the match-up time and manufacturing cost objectives, hence we provide effective exact mathematical programming formulations and heuristic algorithms to provide alternative solutions. Processing time controllability and convex cost functions complicate the scheduling problems significantly. Here, we have also seen that new advances in conic mixed integer programming can play an important role in mitigating this difficulty.

Acknowledgements The authors would like to thank anonymous referees for their helpful guidance on improving the presentation and the contents of the paper. Alper Atamtürk has been supported, in part, by Grant # DMI0700203 from the National Science Foundation. Sinan Gürel was supported by the Scientific and Technological Research Council of Turkey under Grant 2214.

References

Aktürk, M. S., & Görgülü, E. (1999). Match-up scheduling under a machine breakdown. *European Journal of Operational Research*, *112*, 81–97.

Aktürk, M. S., Atamtürk, A., & Gürel, S. (2009). A strong conic quadratic reformulation for machine-job assignment with controllable processing times. *Operations Research Letters*, *37*, 187–191.

Alizadeh, F., & Goldfarb, D. (2003). Second-order cone programming. *Mathematical Programming*, *95*, 3–51.

Bean, J. C., Birge, J. R., Mittenthal, J., & Noon, C. E. (1991). Match-up scheduling with multiple resources, release dates and disruptions. *Operations Research*, *39*(3), 470–483.

Ben-Tal, A., & Nemirovski, A. (2001). *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*. Philadelphia: SIAM.

Bretthauer, K. M., & Shetty, B. (1995). The nonlinear resource allocation problem. *Operations Research*, *43*(4), 670–683.

Briand, C., La, H. T., & Erschler, J. (2007). A robust approach for the single machine scheduling problem. *Journal of Scheduling*, *10*(3), 209–221.

Gürel, S., & Aktürk, M. S. (2007). Optimal allocation and processing time decisions on parallel CNC machines: ϵ-constraint approach. *European Journal of Operational Research*, *183*, 591–607.

Jensen, M. T. (2001). Improving robustness and flexibility of tardiness and total flow-time job shops using robustness measures. *Applied Soft Computing*, *1*, 35–52.

Kayan, R. K., & Aktürk, M. S. (2005). A new bounding mechanism for the CNC machine scheduling problems with controllable processing times. *European Journal of Operational Research*, *167*, 624–643.

Leus, R., & Herroelen, W. (2007). Scheduling for stability in single-machine production. *Journal of Scheduling*, *10*(3), 223–235.

Mehta, S. V., & Uzsoy, R. M. (1998). Predictable scheduling of a job shop subject to breakdowns. *IEEE Transactions on Robotics and Automation*, *14*, 365–378.

Shabtay, D., & Steiner, G. (2007). A survey of scheduling with controllable processing times. *Discrete Applied Mathematics*, *155*(13), 1643–1666.

T'kindt, V., & Billaut, J.-C. (2006). *Multicriteria scheduling: theory, models and algorithms* (2nd ed.). Berlin: Springer.

Vieira, G. E., Herrmann, J. W., & Lin, E. (2003). Rescheduling manufacturing systems: A framework of strategies, policies and methods. *Journal of Scheduling*, *6*, 39–62.

Yedidsion, L., Shabtay, D., & Kaspi, M. (2007). A bicriteria approach to minimize maximal lateness and resource consumption for scheduling a single machine. *Journal of Scheduling*, *10*, 341–352.