Journal of Industrial and Systems Engineering
Vol. 1, No. 4, pp 360-383
Winter 2008

JISE

Study of Scheduling Problems with Machine Availability Constraint

Hamid Reza Dehnar Saidy [1*], Mohammad Taghi Taghavi-Fard[2]

[1]Young Researchers Club, Tehran's Science and Research Branch, Islamic Azad University, Tehran, Iran
Haredes@walla.com

[2]Faculty of Accounting and Management, Allameh Tabataba'i University, Tehran, Iran
dr_taghavifard@yahoo.com

ABSTRACT

In real world scheduling applications, machines might not be available during certain time periods due to deterministic or stochastic causes. In this article, the machine scheduling with availability constraints for both deterministic and stochastic cases with different environments, constraints and performance measures will be discussed. The existing body of research work in the literature will be completely reviewed and the NP-complete models will be identified.

***Keywords**:* Sequencing, Scheduling, Unavailability period, Resumable, Breakdown, NP-hard

1. INTRODUCTION

Scheduling is the process of assigning activities to *resources* over time. At the end, *jobs* are *sequenced* based on a problem performance metric. A variety of constraints such as activity duration, release and due dates, precedence constraints, and resource availability might affect scheduling.

In this article, the effects of resource unavailability on scheduling will be analyzed. Most efforts in the area of scheduling adopt the assumption that machines are continuously available. While this assumption might be justified in some cases, in real world situation, continuous availability of a machine is not usually possible. The machine might not be available due to a deterministic or random cause. This limited availability of machines might result from preschedules, preventive maintenance, or the overlap of two consecutive time horizons in the rolling time horizon planning algorithm. The rolling horizons are used mainly, because most of the real world problems of production planning are dynamic. On the other hand, the input data are being frequently updated. The period in which a machine is not available has been sometimes named a *hole* for convenience (Kubiak et al. 2002).

In section 2, the characteristics and notations of machine scheduling problem under availability constraints will be defined for both deterministic and stochastic cases. In section 3, the existing literature will be reviewed for both cases and finally in section 4, the concluding remarks are presented.

* Corresponding Author

2. PROBLEM CHARACTERISTICS

In production scheduling, orders are fixed in terms of starting and finishing times. When new orders arrive, since there are already orders from the past assigned to machines for processing during various time intervals, they have to be processed using the remaining free processing intervals. The time intervals occupied by the jobs from the previous periods can be thought of as machine unavailable periods for the new jobs to be scheduled. The shift pattern of the facility is another cause for machine unavailability. Workers are obviously needed to operate machines and they cannot be continuously available. They can only work for a certain period of time, for example eight hours a day. According to the number of workers and the demand for the product, the facility can be in operation for a particular period of time in a day. The remaining time can be thought of as unavailable period for the machines. This constraint may be relaxed by increasing the number of shifts. Still another cause for machine unavailability at the beginning of the planning horizon occurs when the machine has to complete the unfinished jobs of the previous planning period. Preventive maintenance is also a reason for machine unavailability. These are examples for which unavailability is initiated by a deterministic cause in the sense that processing times, release and due dates of the jobs and the time and duration of the unavailability period are known at time zero.

Although deterministic models are easier to analyze, they have some drawbacks. Deterministic models assume that there are n jobs (*events*) to be scheduled and that after scheduling these jobs on the resources, the problem is solved. In real world, there might be n jobs in the system at any time, with new jobs being added continuously. Sequencing n jobs has to be done without a perfect knowledge of the near future and the schedule has to cope with unexpected events. One unexpected event might be the entrance of a job with a high priority. This is caused by the operating systems in which subprograms with higher priority interfere with the current ongoing program. Another event which is mostly considered in the models is the breakdown of machines. There may also be delays due to material, changes in release and *tail* dates, tool unavailability, and fluctuations in processing times. All of these events complicate the scheduling problem in most cases. The problem becomes a stochastic scheduling problem when the job processing times, their release dates and the starting and ending times (or duration) of the unavailability periods are not known in advance.

In sub-sections 3.1 and 3.2, the problem characteristics will be presented. Besides, the following notation will be used throughout the article:

- j job index; $j = 1, 2, \ldots, n$ (= number of jobs),
- i machine or resource index; $i = 1, 2, \ldots, m$ (= number of machines),
- J_j job j,
- M_i machine or resource i (in identical processors it is replaced by P_i),
- t_{ij} *processing time* of J_j on M_i; the subscript i is dropped for single machine and parallel machine environments (it is also shown by p_{ij});
- r_j *release date* (*ready time* or head) of the J_j,
- q_j latency duration or tail of operation (or job) j,
- d_j due date (deadline) of J_j,
- w_j weight of job j,
- C_j completion time of job j,
- k number of holes (unavailability periods),
- s_i^k *starting time* of kth unavailability period on M_i; in case there is only one hole, the superscript k is dropped;

e_i^k *ending time* of the *k*th unavailability period on M_i; in case there is only one hole, the superscript *k* is dropped;

S_{ij} *setup time* of the resource *i* on J_j,

n_j number of operations of job *j*,

B_i *beginning time* of the *availability interval* of M_i; the subscript *i* is dropped for single machine;

F_i *finishing time* of the *availability interval* of M_i; subscript *i* is dropped for single machine;

t_j' availability change time point of a certain machine; $0 = t_1' < t_2' < \ldots < t_j' < \ldots < t_Q'$,

$m(t_j')$ number of machines being available during time interval $[t_j', t_{j+1}')$,

C_{max} makespan = $\max\{C_j; j = 1, 2, \ldots, n\}$,

L_{max} maximum lateness = $\max\{C_j - d_j; j = 1, 2, \ldots, n\}$,

t_{max} $\max\{t_j; j = 1, \ldots, n\}$,

$t_{1,max}$ $\max\{t_{1j}; j = 1, \ldots, n\}$,

U_j 1 if job *j* is tardy ($C_j > d_j$) and 0 otherwise,

T_{max} maximum tardiness = $\max\{U_j; j = 1, \ldots, n\}$,

MS_1 $\sum_{j=1}^{n} t_j$,

MS_2 $\sum_{j=1}^{n} (t_{1j} + t_{2j})$,

κ fraction of the semi-processed part that needs to be reprocessed after the machine has become available ($0 \le \kappa \le 1$);

X_j processing requirement of J_j, a random variable,

A_k uptime (a random variable representing the time between (k - 1) and *k*th unavailability periods),

B_k downtime duration (a random variable representing the *k*th unavailability period),

$E[X]$ expectation of random variable *X*.

Throughout this paper the notation used by Pinedo (2001) and Blażewicz et al. (1996) will be adopted with some extensions and modifications. In their notation ($\alpha|\ \beta\ |\gamma$), $\alpha = \alpha_1\alpha_2,\alpha_3,\alpha_4$ denotes the machine (processor) environment, β denotes the problem characteristics and γ denotes the performance measure. Parameter $\alpha_1 \in \{\varnothing, \text{Pm, PF, F, J, O, FF, FJ}\}$ characterizes the processor environment where

$\varnothing$ single machine,

Pm *parallel machine* system (flexible single machine): I or P (*identical* processors), Q (uniform processors), R (unrelated processors),

PF *permutation flow* shop environment,

F *flow* shop dedicated machines system,

J *job* shop dedicated machines system,

O *open* shop dedicated machines system,

FF *flexible flow* shop (multi-processor flow shop, or hybrid flow shop) environment,

FJ *flexible job* shop processor environment.

Parameter α_2 denotes the number of machines or stages in the system ($\varnothing$ if it is arbitrary).

Different patterns of machine availability are "often" discussed for the case of "parallel machine" systems. These are constant, zigzag, decreasing, increasing, and staircase. According to these cases, parameter $\alpha_3 \in \{\varnothing, NC, NC_{zz}, NC_{inc}, NC_{dec}, NC_{inczz}, NC_{deczz}, NC_{sc}\}$ denotes the machine availability for which the following explanations are in order:

(1) If all machines are continuously available, the pattern will be called constant (α_3=Ø).

(2) If there are only n or n - 1 machines available in each interval, then the pattern is called zigzag ($\alpha_3 = NC_{zz}$).

(3) A pattern will be called increasing (decreasing) if for all $j \in IN$: $m(t_j') \geq \max_{1 \leq u \leq j-1}\{m(t_u')\}$ ($m(t_j) \leq \min_{1 \leq u \leq j-1}\{m(t_u')\}$), i.e., the number of machines available in interval $[t_j' - 1, t_j')$ is not larger (smaller) than this number in interval $[t_j', t_j' + 1)$ ($\alpha_3 = NC_{inc}$ (NC_{dec})).

(4) A pattern will be called increasing (decreasing) zigzag if for all $j \in IN$: $m(t_j') \geq \max_{1 \leq u \leq j-1}\{m(t_u') - 1\}$ ($m(t_j') \leq \min_{1 \leq u \leq j-1}\{m(t_u') + 1\}$) ($\alpha_3 = NC_{inczz}$ (NC_{deczz}).

(5) A pattern will be called staircase if for all intervals the availability of M_i implies the availability of M_{i-1} ($\alpha_3 = NC_{sc}$). Patterns (1)-(4) are special cases of (5).

(6) A pattern is called arbitrary if none of the conditions (1)-(5) applies (α_3 = NC). Patterns defined in (1)-(5) are special cases of the one in (6). Some authors use N_{Cwin} instead of NC.

An unavailability period that allows an operation to be interrupted and resumed after a period is called "*crossable*" while an unavailability period that prevents the interruption of any operation, even if the operation is *resumable*, is called "*non-crossable*". We distinguish three cases: all unavailability periods are *crossable*, denoted by *cr*; all unavailability periods are *non-crossable*, denoted by *ncr*; some unavailability periods are *crossable* and some *non-crossable*, denoted by *cr/ncr* ($\alpha_4 \in \{cr, ncr, cr/ncr, \varnothing\}$).

Most of the models with availability constraints are derived without considering α_3 condition and are applicable under conditions (2)-(6). In the literature, there are 4 cases relevant to this subject matter. When a job cannot be finished before the next down period of a machine and the job has to restart, then the job is called *nonresumable* (*nr*). If a job has to partially restart after the machine has become available, then it is called *semiresumable* (*sr*). If a task can continue to be processed on the same machine after the machine has become available, then the job is called *resumable* (*rs*). In this case, some authors say that pre-emption is allowed. If some operations are *resumable* and some others *non-resumable*, availability constraint will be denoted by *rs/nr*. β belong to {n = b, *pmtn*, *ppmtn* , *prec*, r_j, *chains*, *tree*, *no-wait*, *rs*, *sr*, *nr*, *rs/nr*, (M^k_i), d_j, S_{ij}, t_j = p, Ø}; b and p are constant definite numbers and M^k_i shows that there are k holes on M_i, and if i isn't defined by a constant number, it means k unavailability periods exist on all processors. When there is only one hole, the superscript k is dropped. Some authors use $h_{k,j}$ instead of M^k_i. The *pmtn* (*prmp*), *ppmtn* and *prec* indicate the preemption, partial pre-emption and general precedence in operations, respectively.

3. LITERATURE REVIEW

This work intends to collect all published cases (work issues) in the literature up to this date; hence, all models and problems - which have been cited or published with any cases, environments, constraints, characteristics and performance measures - are cited here. Note that if a single machine problem or model is NP-complete its extension to PF, F, J, O, FF and/or FJ environment will also be NP-complete and this rule is valid for other similar extensions. Furthermore, if a problem or model is NP-hard in deterministic case, it will be more complicated in stochastic case. Moreover, a problem will be NP with availability constraints if it is NP without those constraints.

3.1. Deterministic Case

In deterministic models, the time and the duration of the unavailability period is known in advance. It is assumed that M_i is unavailable during the period from s_i to e_i ($0 \leq s_i \leq e_i$) when there is at most one unavailability period. If there is more than one unavailability period, then the kth unavailability period on M_i will be started and finished at time s_i^k and e_i^k, respectively. In the case of preventive maintenance, generally at most one unavailability period is assumed; because it is unlikely that we have more than one preventive maintenance period on the shop floor during the scheduling horizon. Also, at most one unavailability period is assumed for the special case where the machine may not be available at the beginning of the planning horizon ($s_i = 0$) because of the uncompleted jobs scheduled in the previous planning period. In the shift pattern of the facility, the number of unavailability periods may be more than one in the planning horizon.

Each job has a known processing requirement of t_{ij} time units. Each machine may process only one job and each job may be processed by only one machine at a time. The aim will be to find a feasible schedule for n jobs, if one exists, such that all jobs can be processed within the given intervals of the machine availability while optimizing a performance criterion. The performance measures studied in the literature include minimizing the *makespan*, *flow time*, *lateness*, L_{max}, T_{max}, $\sum C_j$, *weighted sum of completion times* ($\sum w_j C_j$), *number of tardy jobs* ($\sum U_j$), *weighted sum of the number of tardy jobs* ($\sum w_j U_j$) and *greatest completion time of an operation* ($max_{1 \leq j \leq n}(C_j + q_j)$) as well as maximizing *node (resource) availability* and *minimum completion time*. The last objective is used for balancing machines utilization levels.

Machine scheduling problems with availability constraints have been studied in the literature in the single machine, parallel machines, flow shop, job shop, open shop, flexible flow shop and flexible job shop environments. Some problems can be solved optimally by extending the classical algorithms used for the problems where the machines are continuously available. Most problems are NP-hard. For problems in class-NP *dynamic programming* (DP) algorithms have been proposed for finding optimal solution(s) and/or heuristics with error bound analysis. To show the efficiency of the proposed algorithms three types of error bounds are used. Let f_H be the objective function value obtained by applying some proposed heuristic H for the problem and f^* be the objective function value of the optimal schedule. Then $A = f_H - f^*$ is the absolute error, $R_H = (f_H - f^*)/f^*$ is the relative error and f_H / f^* is the performance ratio (Türkcan 1999).

3.1.1. Single machine problems

3.1.1.1. Nonresumable availability constraints

Lee (1996) showed that the problem 1| *nr* |C_{max} is NP-hard when single (or multiple) period(s) of unavailability (maintenance) occurs. The Longest Processing Time (LPT) algorithm has a relative tight error bound of 1/3. Adiri et al. (1989) and Lee and Liman (1992) have studied the problem 1|*nr*|$\sum C_j$. They have shown that the problem is NP-hard. Lee and Liman (1992) have shown that the SPT algorithm has a worst case relative error bound of 2/7 which is tight. Lee (1996) has shown that the problems 1| *nr* |L_{max}, 1| *nr* |$\sum U_j$ and 1| *nr* |$\sum w_j C_j$ are also NP-hard. When the problem 1|*nr*|L_{max} is solved by EDD rule, the error bound is t_{max} and when the Moore-Hodgson's algorithm (1968), MH, is used to solve the problem 1| *nr* |$\sum U_j$, the error bound is 1. For the problem 1|*nr*|$\sum w_j C_j$, the performance ratio of SWPT might be arbitrarily large even while $\forall j$: $w_j = t_j$.

The problem 1| *nr* |$max(C_j + q_j)$ was initially solved by Leon and Wu (1992) by a branch and bound (B&B) algorithm that can solve problems with up to 50 operations. Later, Balas et al. (1998) considered the single machine problem with delayed precedence constraints and deadlines.

For *batch production* on a single machine, Wang and Cheng (2006) proposed a heuristic with a worst-case error bound of 1/2 and showed that this bound is tight. Of course, they considered both production and job delivery at the same time where the objective is to minimize the *arrival time of the last delivery batch to the distribution center* (this metric is equivalent to C_{max}). They also assumed that one vehicle with at most *K*-job capacity is available to deliver the jobs in a fixed transportation time to a distribution center.

Gawiejnowicz (2007) has presented an algorithm to minimize C_{max} for sequencing n deteriorating jobs on a single machine time-dependent scheduling with non-availability periods (TDSNP) problem and k holes (disjoint periods); $1 \leq k < n$ and $s_1 > 0$. In this problem, $t_j = \alpha_j.t$, where $\alpha_j > 0$ is deterioration rate and $t > 0$ is time (t is applied instead of the starting process time of J_j). He has proved that if $k = 1$, TDSNP problem is NP-complete in the ordinary sense; else ($k > 1$), TDSNP problem is NP-complete in the strong sense.

Chen (2007) has considered a periodic maintenance scheduling problem on a single machine in a textile company. A periodic maintenance schedule consists of several maintenance periods and each maintenance period is scheduled after a periodic time interval. For simplicity, the processing times and due dates can take only integral values. He has developed a near optimal heuristic and an optimal B&B algorithm to minimize the performance measure T_{max}. The computational results show that the proposed heuristic is highly accurate and efficient (quick).

3.1.1.2. Resumable availability constraints

Lee (1996) studied the single machine problem for different performance measures. He showed that the makespan for a single machine problem with resumable availability constraint ($1|\ rs\ |C_{max}$) is minimized by an arbitrary sequence. The minimization of flow time with resumable availability constraint on a single machine problem ($1|\ rs\ |\sum C_j$) is solved optimally by the Shortest Processing Time (SPT) algorithm. In SPT, jobs are scheduled in a non-decreasing order of their processing times. $1|\ rs\ |L_{max}$ can be solved optimally by the Earliest Due Date (EDD) algorithm, where the jobs are scheduled in a nondecreasing order of their due dates. He showed that the MH algorithm can be modified to solve the problem $1|\ rs\ |\sum U_j$ optimally in $O(n.logn)$ time.

We know that the problem $1|\ |\sum w_jC_j$ can be solved optimally by the Shortest Weighted Processing Time (SWPT) rule; but when availability constraint is added, Lee (1996) has shown that the problem $1|\ rs\ |\sum w_jC_j$ becomes NP-hard, even if $w_j = t_j$ for all j. A DP algorithm of $O(n.t_{max}.s_1)$ - which solves the problem with single unavailability period optimally - is provided by Lee (1996). Heuristics with error bound analysis have also been proposed for this problem. The error bound of SWPT rule is $w_j(e_1 - s_1)$. The performance of SWPT algorithm might be arbitrarily large even while $w_j = t_j$ for all j. Lee (1996) proposed a heuristic to solve this problem when $w_j = t_j$ with a relative error bound of 1.

Wu and Lee (2003) submitted an algorithm to minimize C_{max} for scheduling linear deteriorating jobs on a single machine for a for a TDSNP problem with 1 availability constraint.
The model $1,\alpha_1|\ r_j, \beta_1(M^k_1)\ |max(C_j + q_j)$ with $\beta_1 \in \{rs, nr, rs/nr\}$ is strongly NP-hard since the same model without unavailability periods is strongly NP-hard. Canon et al. (2003) solved the problem $1|rs|max_{1 \leq j \leq n}(C_j + q_j)$ by a simple adapting Carlier's B&B algorithm (1982). Problems with up to 500 jobs are solved in less than 1 minute in the worst case and in less than 1 second on the average by this method. The problem $1,cr/ncr|rs|max(C_j + q_j)$ can be efficiently solved by embedding a B&B method for $1|\ rs\ |max(C_j + q_j)$ in the solution method of $1|\ nr\ |max(C_j + q_j)$ (Mauguière et al. 2005).

For minimizing the arrival time of the last delivery batch to the distribution center (~ C_{max}) in batch production on a single machine, Wang and Cheng (2006) provided a polynomial algorithm to solve the problem optimally.

3.1.1.3. Resumable/ Nonresumable availability constraints

For the problem 1,*cr*| *rs/nr* |*max*(C_j + q_j), a B&B algorithm was proposed by Mauguière et al. (2003a), which makes it possible to solve most of instances with up to 100 operations, though some smaller instances seem to be intractable for the procedure.

Mauguière et al. (2003b) solved the problem 1,*cr/ncr*| r_j, *rs/nr*(M^k_1), d_j |*max*(C_j + q_j) by B&B procedure. Another solution method has been proposed by Mauguière et al. (2005). They also solved the strongly NP-hard problem 1| *ppmtn*, r_j, d_j, q_j |*max*(C_j + l_j) by means of an approximation algorithm which is a modification of Schrage's algorithm (Schrage 1971). In this problem the latency duration for job *j*, l_j, is equal to max {q_j, K – d_j}, where K is a constant number. The "*rs*", "*cr/ncr*| *rs*", "*nr*", "*cr*| *rs/nr*" and "*ppmtn*" problems are subsets of "*cr/ncr*| *rs/nr*" problem and so, 1,*cr/ncr*|r_j,*rs/nr*, d_j |*max*(C_j + q_j) is too complicated and hard, but tests have shown that the algorithm proposed by Mauguière et al.'s algorithm (2005) solves it in favorite time and accuracy.

3.1.2. Parallel machine problems

3.1.2.1. Nonresumable availability constraints

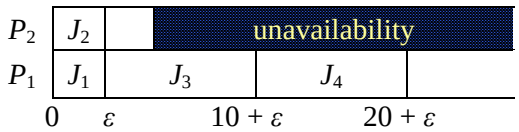
Mosheiov (1994) studied the problem Pm| *nr* |$\sum C_j$ where M_i is available in time interval [B_i, F_i] and showed that SPT is asymptotically optimal as the number of jobs goes to infinity. Lee (1996) has shown that the problem Pm2| *nr* |$\sum w_j C_j$ is also NP-hard. When w_j = 1 for all *j* and M_1 is available all the time, he provides a DP algorithm of $O(n.MS_1.s_2)$ to solve the problem optimally. Moreover, Lee and Liman (1993) have studied the same problem where machine P_1 is available all the time and the "identical" machine P_2 is available from time 0 to a fixed point in time and have shown that the problem is NP-hard. They provided a DP algorithm and proposed a modified SPT based heuristic with a worst case error bound of 1/2. The modified SPT rule has the following form:

Step 1: Assign the shortest task to P_1.

Step 2: Assign the remaining tasks in SPT order alternately to both machines until no other tasks can be assigned to P_2 without violating F_2.

Step 3: Assign the remaining tasks to P_1.

Figure 1 illustrates how that bound can be reached asymptotically (when ε tends toward 0). The modified SPT rule leads to a large idle time for machine P_1.



$\sum C_j = 30 + 4\varepsilon$, optimum is $20 + 5\varepsilon$ ($t_1 = t_2 = \varepsilon$, $t_3 = t_4 = 10$)

Figure 1- Example for the modified SPT rule from Lee and Liman (1993).

Ullman (1975) was the first who studied the problem I,NC| $|C_{max}$. Lee (1996) has shown that this problem is strongly NP-hard (3-partition is a special case). The List Scheduling (LS) algorithm has an error bound of m and the LPT algorithm has a tight error bound of $(m + 1)/2$. If machines have different beginning times B_i, the LPT rule will lead to a relative error of $R_{LPT} \leq 1/2 - 1/(2m)$ or of $R_{MLPT} \leq 1/3$ if the rule is appropriately modified (see Lee 1991). Both bounds are tight. Note that a LPT algorithm leads to a relative error of $R_{LPT} \leq 1/3 - 1/(3m)$ for continuously available machines (see Graham 1969). The modification uses dummy tasks to simulate the different machine starting times B_i. For each machine P_i, a task T_j with processing time $t_j = B_i$ is inserted. The dummy tasks are merged into the original tasks set and then all tasks are scheduled according to the LPT rule under an additional restriction that only one dummy task is assigned to each machine. After finishing the schedule, all dummy tasks are moved to the head of machines followed by the remaining tasks assigned to each P_i. The LPT rule runs in $O(n.logn)$ and MLPT in $O((n + m)log(n + m) + (n + m)m)$ time. Kellerer (1998) presented a dual approximation algorithm using a bin packing approach which leads to a tight bound of 1/4 (see Sanlaville and Schmidt 1998). Also for $m = 1$ the problem remains NP-complete as demonstrated by Lee (1996).

Unit Execution Time and Arbitrary Precedence Constraints

Unit execution time (UET) scheduling is for cases that all jobs (tasks) have equal operation time which can be assumed to be equal to 1. This is important in application for two reasons: first, it contains several frontier problems when looking at complexity issues, and second, it models a restrictive version of preemption, when interrupting a task is only allowed at specified [integer] moments. Of course, availability changes are also restricted to integer moments.

The considered models are applicable to single machine problems with non-resumable availability constraints as well. Based on the performance metric, we divide this case into 2 sections *A* and *B.*

A. Minimizing the maximum lateness

Brucker et al. (1977) proved that an Earliest Due Date (EDD) rule can be applied to modified due dates and is optimal if the precedence graph is an in-tree. Liu and Sanlaville (1995a) proved that this remains true, with a similar modification scheme, for increasing zigzag availability patterns. In the same way, Garey and Johnson (1977) proposed an off-line modification scheme so that EDD builds optimal schedules on two machines, for arbitrary task graphs. This result can also be extended to arbitrary availability patterns as it has been shown by Liu and Sanlaville (1995b).

B. Minimizing the makespan

Problem with arbitrary precedence constraints is NP-complete even for a constant (continuous) availability pattern. If the precedence graph is an inforest, the problem is still NP-complete for a decreasing pattern. Dynamic programming algorithms might be used for this case.

Some list algorithms are optimal for some specific availability patterns (see Sanlaville and Schmidt 1998). The list algorithm of Coffman and Graham (1972) is optimal for two machines with arbitrary precedence constraints and an arbitrary pattern (see Liu and Sanlaville 1995a). If the graph is an interval order graph, the list algorithm will choose the first tasks with the largest set of successors (Most Successor First (MSF) rule) that is optimal on an arbitrary availability pattern (see Liu and Sanlaville 1995b). Figure 2 shows an instance where any other choice for the first task to be scheduled leads to a sub-optimal solution (MSF priority list is J<2-1-3-5-6-4-7> for single and parallel machine). Interval order graphs attracted much attention as any precedence graph might be

transformed to an interval order graph by adding a set of precedence relations (refer to Papadimitriou and Yanakakis, 1979). Hopefully, optimal schedules for interval order graphs might lead to satisfactory schedules for arbitrary task graphs.

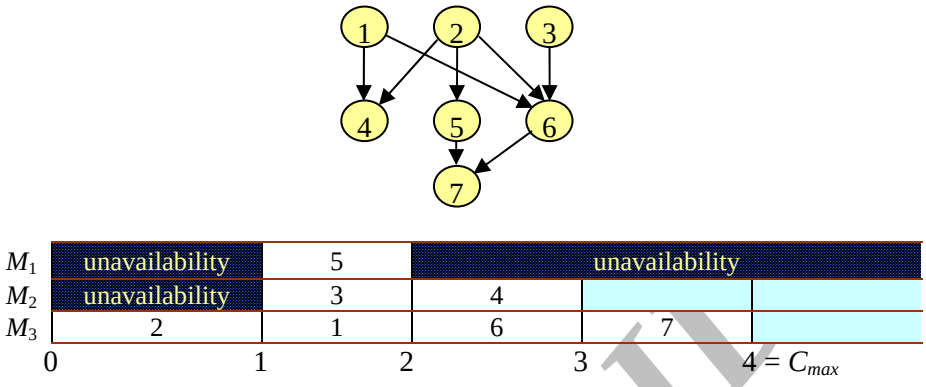


Figure 2- Use of the MSF rule for interval order graphs.

Dolev and Warmuth (1985a) demonstrated that Highest Level First (HLF) schedules are optimal if the precedence graph is either an inforest (outforest) and the pattern is NC_{inczz} (NC_{deczz}), or forms chains of tasks and arbitrary patterns (see Liu and Sanlaville, 1995a). Dolev and Warmuth (1985b), then showed that HLF policy leads to an $O(n.logn)$ "flip-flop" algorithm for scheduling opposing forests to zigzag patterns with 2 or 3 machines. Opposing forests are the union of out-trees and in-trees as hinted by Sanlaville and Schmidt (1998).

3.1.2.2. Resumable availability constraints

Lee (1996) has stated that the problem Pm| *rs* |C_{max} which is harder than Pm| |C_{max}, is NP-hard. Lee (1991) studied the problem when the unavailability period is at the beginning of the planning horizon and there is at most one unavailability period ($s_i = 0$, $e_i \geq 0$ for all i). The LPT rule has a tight error bound of 1/2 and the modified LPT algorithm has a tight error bound of 1/3 for it. Lee (1996) has provided an error bound analysis for the algorithms LPT1 and LPT2. LPT1 algorithm assigns jobs to the minimum loaded machine. In LPT2 algorithm, J_j is assigned to a machine such that its finishing time is minimized. The performance ratio of LPT1 can be arbitrarily large even for $m = 2$ when $s_i > 0$ for all i. The R_{LPT2} bound is $^{(m - 1)}/_{2m}$ which is a tight bound. Since $1|rs|\sum w_jC_j$ is NP-hard, Pm2| *rs* $|\sum w_jC_j$ is also NP-hard for $n > 2$. This problem is studied by Kaspi and Montreuil (1988). When the unavailability period occurs at the beginning of the planning horizon, SPT gives the optimal schedule. Lee (1996) provided a Dynamic Programming approach of $O(n.MS_1.s_2.t_{max})$ which solves the problem optimally. Lin et al. (1998) studied the parallel machine problem when the unavailability period is at the beginning of the planning horizon and there is at most one unavailability period. The objective is the maximization of the "minimum completion time". They showed that LPT has a worst case error bound of $(2m - 1)/(3m - 2)$.

Schmidt (1984) studied the problem Pm| *prmp*, *rs* |C_{max} and gave the conditions for the existence of a feasible preemptive schedule when all machines are available in an arbitrary number of time intervals. Such a feasible schedule can be constructed in $O(n + m.logm)$ time. He showed that the number of induced preemptions is proportional to the total number of processing intervals of all processors. Later, Schmidt (1988) considered a more generalized problem which takes into account different release and due dates and can be solved in $O(n.m.logm)$ time. For this model, if all machines are only available in one and the same time interval (B, F) and tasks are independent,

McNaughton (1959) has shown that there is a feasible preemptive schedule iff $\max_j\{t_j\} \leq (F\text{-}B)$ and $\sum_j t_j \leq m(F\text{-}B)$. He gave an algorithm of $O(n)$ with at most $m - 1$ preemptions to construct this schedule.

Lawler and Martel (1989) solved the problem Q2| *pmtn*, *r*s $|\sum w_j U_j$. They used dynamic programming to propose pseudo-polynomial algorithms ($O(\sum_j w_j.n^2)$ or $O(n^2.t_{max})$). Nothing, however, was say about the effort needed to compute processing capacity in one interval.

For the problem I| *pmtn*, *r*s $|L_{max}$, Sanlaville (1995) suggested a nearly on-line priority algorithm with an absolute error of $A \leq (m - {}^1/_m)t_{max}$ if the availability of machines follows a constant pattern, and of $A \leq t_{max}$ if the machine availability refers to an increasing zigzag pattern. The priority is calculated according to the Smallest Laxity First (SLF) rule, where laxity (or slack time) is the difference between the task's due date and its remaining processing time. The SLF algorithm executes in $O(n^2.t_{max})$ and it is optimal in the case of a zigzag pattern and no release dates. Also for I| *pmtn*, r_j , *r*s , d_j $|L_{max}$, if the number of changes of machine availabilities during any time interval is linear in length of the interval, an algorithm can be implemented in $O(n^3.t_{max}{}^3(logn + logt_{max}))$ (see Sanlaville 1995).This algorithm needs the knowledge of all the data at time 0 and hence is off-line. When no release dates are given but due dates have to be considered, L_{max} can be minimized using the approach suggested by Schmidt (1988) in $O(n.m.logn)$ time. He has showed that the number of induced pre-emptions is proportional to the total number of processing intervals and deadlines.

Liu and Sanlaville (1995a) studied the parallel machine problem with resumable availability constraints considering the precedence constraints. Problems with chains and arbitrary pattern of unavailability can be solved in polynomial time by the Longest Remaining Path (LRP) rule for minimizing the makespan (Pm,NC| *prmp*, *chains* $|C_{max}$). In case of two parallel machines and arbitrary patterns of availability, LRP solves problems with arbitrary task precedence relations (Pm2,NC| *prmp*, *prec* $|C_{max}$) in time complexity and number of preemptions of $O(n^2)$.

Liu and Sanlaville (1995a) showed that results on minimization of C_{max} for inforest precedence graphs and increasing zigzag patterns can be extended to minimization of L_{max}, using SLF rule on modified due dates (the modified due date is given by $d'_j = min\{d_j, d_{s(j)} + t_{s(j)}\}$, where index $s(j)$ is related to successor job of the J_j when it exists). In the same way, minimizing L_{max} on two machines with availability constraints is achieved using SLF with a different modification scheme.

Albers and Schmidt (1999, 2001, 2004) investigated an online version of a basic problem P,NC|*pmtn* $|C_{max}$ and presented an online algorithm that constructs schedules with an optimal makespan if a lookahead period of one unit is given, i.e., the algorithm always knows the next point in time when the set of available machines changes. Also, they gave an online algorithm without lookahead that constructs nearly optimal schedules. They showed that no online algorithms can construct optimal schedules and online algorithms can achieve a bounded competitive ratio if there are time intervals during which no machines are available. Schmidt (2000a) also presented a paper on performance guarantee of two simple priority rules for offline and online production scheduling with limited machine availability.

The problem of scheduling n preemptive jobs on m machines with identical speed under machine availability and "eligibility" constraints for minimizing L_{max} has been considered by Sheen and Liao (2007). Network flow technique is used to formulate this scheduling problem into a series of maximum flow problems. They have proposed a polynomial time two-phase binary search algorithm to verify the feasibility of the problem and to solve the scheduling problem optimally if a feasible schedule exists. Finally, they show that if x is the total number of availability intervals on

all machines, and *UB* and *LB* are upper and lower bounds found for optimal L_{max} by their proposed algorithm, respectively; the time complexity of the algorithm will be $O((n+(2n+2x))^3 log(UB - LB))$.

Blażewicz et al. (2000, 2003) investigate preemtable tasks and multiprocessor tasks on parallel processors with limited availability and show that this problem becomes NP-hard in the strong sense in case of trees and identical processors. If tasks form chains and also they are processed by identical processors with a staircase pattern (NC_{sc}) of availability, the problem can be solved in low-order polynomial time for criterion C_{max} and a linear programming approach is required for criterion L_{max}. The network flow and linear programming approaches are proposed for independent tasks scheduled on uniform (Q) and unrelated (R) processors with arbitrary patterns of availability for schedule length and maximum lateness criteria, respectively.

Leangsuksun et al. (2005) proposed concepts of integrating high availability cluster mechanism with a secure cluster infrastructure. In high-availability problems, the resource (computer or node) availability is very important to optimize overall performance. Apon and Wilbur (2003) designed an advanced multi processor network (AmpNet) with a high availability in mind.

In batch production on two parallel machines while only one processor has an unavailable interval, Wang and Cheng (2006) proposed a heuristic to minimize the arrival time of the last delivery batch to the distribution center with a worst-case error bound of 2/3.

Blażewicz et al. (1996) edited a book about this field that can be regarded for detailed studying.

3.1.3. Flow shop and permutation flow shop problems

As demonstrated in Kubiak et al. (2002) and Aggoune and Portmann (2006), the problem F| r_j, nr |Z in which Z is one of the performance measures mentioned in this paper for $m \geq 2$ is strongly NP-complete; hence, most researchers have just presented algorithms for 2 machines.

3.1.3.1. Nonresumable availability constraints

As stated in section 3.1.1.1, based on the work of Adiri et al.'s work (1989), the problem F| nr |$\sum C_j$ (and so, F| $nr(M^k_i)$ |$\sum w_j C_j$) is NP-complete, because the problem 1| nr |$\sum C_j$ just by one unavailability period is NP-complete. The SPT rule leads to a tight relative error non-greater than 2/7 for this problem. For fixed m, the SPT rule is asymptotic optimal if there is not more than one interval of non-availability for each machine (refer to Sanlaville and Schmidt 1998).

Allahverdi (1996) considered a two-machine flow shop problem and showed that if only the first machine breaks down, the LPT policy will minimize the maximum lateness; while if only the second machine breaks down, the SPT policy must be used.

Cheng and Wang (1999) studied the problem F2| $nr(M_i^2)$ |C_{max} with two consecutive availability constraints. They developed a heuristic and showed that it has a worst-case error bound of 2/3.

Lee (1999) studied the two-machine flow shop problem with nonresumable availability constraints. He proposed a heuristic with a relative error bound of 1 when the availability constraint is imposed on machine 1. When the availability constraint is imposed on machine 2, Johnson's Algorithm, JA, (Johnson 1954) has a tight relative error bound of 1. Braun et al. (2002) surveyed the problem of minimizing the makespan in the two-machine n-job flow-shop scheduling with k_1 non-availability intervals on each of the two machines. This problem is binary NP-hard even if there is only one

non-availability interval either on the first machine or on the second machine. Aggoune and Portmann (2006) proved that the problem F,N_{Cwin}| $n = 2$ |C_{max} is polynomial and its complexity by using their graphic method is at most equal to $O(k.m^4)$.

Ng and Kovalyov (2003) studied a two-machine flowshop scheduling problem with an assumption that one of the two machines is not available in a specified time period. The problem is known to be NP-hard. Pseudo-polynomial dynamic programming algorithms and heuristics with worst case error bounds are given in the literature to solve the problem. Those are different for cases the unavailability interval is for the first or the second machine. The existence of a fully polynomial time approximation scheme (FPTAS) was formulated as an open conjecture in the literature. In this paper, it has been shown that the two cases of the problem under study are equivalent to similar partition type problems; then, authors derived a generic FPTAS for the latter problem with $O(n^5/\varepsilon^4)$ time complexity. Espinouse et al. (1999) solved the two-machine no-wait flow-shop problem for minimizing maximum completion time by assuming each machine has one hole. Wang and Cheng (2001) provided $^5/_3$-approximation algorithms for this problem with an unavailable interval. Cheng and Liu (2003a, 2003b) surveyed the approximation methods for this problem, too. Since the problem with an unavailable interval is NP-hard and the problem with two separate unavailable intervals has no polynomial time approximations with constant performance bounds unless P = NP (see Espinouse et al. 1999), they have presented a polynomial time approximation scheme for the problem when the unavailable interval is imposed on only one machine, or the unavailable intervals overlap on the two machines.

Aggoune (2004b) presented a heuristic based on the genetic algorithm and taboo search for the problem F| nr |C_{max}. Afterwards, Aggoune and Portmann (2006) proposed a new heuristic algorithm for this problem based on the development of Aggoune' heuristic for 2-job (Aggoune 2004a). It consists of a procedure that schedules jobs two by two following an input sequence, combined with a tabu search (TS). Aggoune and Portmann's algorithm is more exact than Aggoune's algorithm (2004b), but their algorithm needs more time to access the schedule. The problem F,N_{Cwin}| |C_{max} is NP-hard in the strong sense. It has been shown by Kubiak et al. (2002) that it is impossible to find a heuristic with performance guarantee for the makespan minimization in a two-machine flow shop, if more than one unavailability period is considered on each machine.

3.1.3.2. Semiresumable availability constraints

Lee (1999) has shown that the two-machine flow shop with semi-resumable availability constraints imposed on both machines is NP-hard even if $s_1 = s_2 = s$ and $e_1 = e_2 = t$. He provided a pseudo-polynomial DP algorithm of $O(n.MS_2^2.s_1.\ t_{1,max})$ for the problem F2| $sr(M_1)$ |C_{max}. The JA has a tight relative error bound of 1 for this problem. Indeed, the JA has a tight relative error of $max\{1/2, \kappa\}$ and it is optimal when $s_1 = s_2 = 0$ and has a relative error bound of κ when $s_1 = s_2 = s$ and $e_1 = e_2 = t$.

3.1.3.3. Resumable availability constraints

Lee (1977) studied the two-machine flow shop model with availability constraints imposed on only one machine. He showed that the problems F2| $rs(M_1)$ |C_{max} and F2| $rs(M_2)$ |C_{max} are both NP-hard in the ordinary sense. He also provided a DP algorithm of $O(n.MS_2^2.s_1.t_{1,max})$ to solve the problem F2|$rs(M_1)$|C_{max} optimally. The JA - which solves F2| |C_{max} optimally - has a relative error bound of 1 and a heuristic was proposed by him with a relative error bound of 1/2 based upon the JA for the above model. The JA has a relative error bound of 1/2 for the problem F2| $rs(M_2)$ |C_{max}. He proposed a heuristic to solve this problem with a relative error bound of 1/3. In addition, he proposed heuristics with one hole on *either* M_1 or M_2 with relative errors 3/2 and 4/3, respectively!

Lee (1999) studied this problem again. He proved if $s_1 = s_2 = s$ and $e_1 = e_2 = t$, then the JA will solve the problem optimally. He showed that the problem is NP-hard when $s_1 \neq s_2$ and $e_1 - s_1 = e_2 - s_2$. In this case, the relative error bound of JA might be arbitrarily large. Breit (2004) studied this problem, too. He presented an improved algorithm with a relative worst-case error bound of 5/4 while the best "fast" approximation algorithm for this problem guarantees a relative worst-case error bound of 4/3. The tight worst-case bound for the problem F2| $rs(M_2)$ $|C_{max}$ must be ¼. Kubiak et al. (1997) proposed a B&B algorithm for a 2-machine problem, too.

The above results suggest that if there is to exist 1 hole in F2| $rs(M_i)$ $|C_{max}$, it will be better for heuristics that it occurs on M_2; but Kubiak et al. (2002) showed it isn't generable when at least two holes are allowed to occur, unless P = NP. In other words, they proposed a simple heuristic based on JA that guarantees a relative error of 2 in $O(n.logn)$ time if all holes occur on M_1; but, such holes on M_2 make polynomial time heuristics with constant relative error impossible, unless P = NP. The optimum sequence between each 2 holes is Johnson's sequence. They developed a branch and bound heuristic based on the above property for F2| $rs(M^k_i)$ $|C_{max}$, too. In addition, their tests show that there is no significant difference among computational times for sample examples if holes are allowed to occur at least on one of the two machines. There is no polynomial time heuristics with a relative constant error for F2| $rs(M^2_i)$ $|C_{max}$. The problem F2| $rs(M_i)$ $|C_{max} \leq y$ is usually NP-Complete (see Blazewicz et al. 2001). Schmidt (2000b) proposed a parallel B&B algorithm for this problem. For more details, we refer to the survey of Schmidt (2000b), where existing methods for solving scheduling problems under availability constraints as well as complexity results are reviewed.

Cheng and Wang (2000) showed that the worst-case error bound 1/2 of the heuristic provided by Lee (1997) for the problem F2| $rs(M_1)$ $|C_{max}$ is tight and then, they developed an improved heuristic with a worst-case error bound of 1/3. Wang and Cheng (2007a) have propounded 2 heuristics for the problem F2| $rs(M_i)$, S_{ij} $|C_{max}$ and show that their worst case error bounds are no longer than 2/3.

Wang and Cheng (2007b) have studied the problem PF2| $rs(M_1)$, S_{ij} $|C_{max}$ and presented a polynomial-time approximation scheme for it.

3.1.4. Job shop problems

3.1.4.1. Nonresumable availability constraints

The single machine algorithm of Balas et al. (1998) is used in a Shifting Bottleneck Procedure to solve the job-shop scheduling problem with deadlines. This algorithm can also be used to give an approximation algorithm for the problem J| nr, d_j $|max(C_j + q_j)$ (see Carlier 1982). The problem J|nr|C_{max} was solved exactly by Aggoune (2002) using a B&B algorithm with lower bound based on a two-job-shop problem with heads and tails and unavailability periods.

Aggoune (2004a) extended Akers' geometric approach to J| $nr(M^k_i)$, $n = 2$ $|C_{max}$ and named his method *temporized geometric approach* (*TGA*) that is polynomial and its complexity is at most equal to $O(k.u^4)$, where $u = max\{n_1, n_2\}$.

3.1.4.2. Resumable availability constraints

Mauguière et al. (2003a) proposed a B&B algorithm for the problem J| $rs(M^k_i)$ $|C_{max}$. Computational results show that solving the problem J| rs $|C_{max}$ is a little more difficult than the problem without unavailability periods.

Aggoune (2004a) extended his nonresumable "*TGA*" to J| $rs(M^k_i)$, $n = 2$ |C_{max} that is polynomial and its complexity is at most equal to $O(ku^4)$; $u = max\{n_1, n_2\}$.

3.1.4.3. Resumable/ Nonresumable availability constraints

Mauguière et al. (2003b) proposed a B&B algorithm for the problem J,*cr*| *rs/nr*(M^k_i) |C_{max}. They have extended their job-shop algorithms to J,*cr/ncr*| r_j, *rs/nr*(M^k_i) |C_{max} afterward (Mauguière et al. 2005).

3.1.5. Open shop problems

3.1.5.1. Nonresumable availability constraints

Breit et al. (2001a, 2001b) studied a two-machine open shop scheduling problem without preemption, in which one machine is not available for processing during a given time interval. The objective is to minimize the makespan. They showed that the problem is NP-hard and presented an approximation algorithm with a worst-case ratio of 4/3.

3.1.5.2. Resumable availability constraints

Lorigeon et al. (2002) studied O2| *rs* |C_{max}. This problem is NP-hard. They developed a dynamic programming algorithm with pseudo-polynomial time to solve the problem optimally when a machine isn't available at time $s_i > 0$. Then, they proposed a mixed integer linear programming formulation that allows solving instances with up to 500 jobs optimally in less than five minutes with CPLEX solver. Moreover, they showed that any heuristic algorithm has a worst-case error bound of one for this problem. Kubzin et al. (2005) considered the problem O2| *rs* |C_{max}, as well. They presented two polynomial-time approximation schemes: one of which handles the problem with one non-availability interval on each machine and the other for the problem with several unavailability intervals on one of the machines. Problems with a more general structure of the unavailability intervals cannot be approximated in polynomial time within a constant factor, unless P = NP.

3.1.6. Flexible flow shop

Allaoui and Artiba (2006) surveyed the two-stage hybrid flow shop non-resumable scheduling problem to minimize C_{max} with only one machine on the first stage and m machines on the second stage. They considered that each machine is subject to at most one unavailability period and discussed the complexity of the problem and proposed a B&B model for its solution. Finally, they have calculated the worst-case performances of three heuristics: LIST algorithm, LPT algorithm and H-heuristic.

Jungwattanakit et al. (2007) have formulated a 0-1 mixed integer program for minimizing the convex combinations of C_{max} and $\sum U_j$ in FF problem scheduling with n independent jobs, unrelated parallel machines in each stage, release dates, due dates, sequence- and machine-dependent setup times, which is often present in the textile industry. They assumed that only one unavailability period *may* occur for each machine in zero time ($s_i = 0 \leq e_i$) and the preemption of jobs isn't permitted (the problem is non-resumable). Since this problem is NP-hard in the strong sense, they have developed heuristic algorithms to solve it approximately. Initially, several basic existing dispatching rules and well-known constructive heuristics for flow shop makespan scheduling problems were extended to solve the problem under consideration. To improve the solutions,

polynomial heuristic improvement methods based on the shifting of the jobs were then applied. Finally, genetic algorithms were suggested to tackle this problem.

3.1.7. Flexible job shop (FJ)

Levitin (2000) provided some details about the flexible job-shop sequencing and scheduling problems. A genetic algorithm-based approach was developed by Chan et al. (2006) for assigning operations to machines and sequencing jobs on machines to optimize the system objectives in FJ resource-constrained problem iteratively.

Dehnar Saidy and Taghavi-Fard (2008) propounded an exact geometric algorithm for the problem $FJ,cr/ncr|\ n = 2,\ rs/nr/sr(M_i^k),\ r_j,\ m_i \leq 2,\ S_{ij}\ |\gamma$, where m_i is the number of identical processors in each stage (work center) and γ is one of the performance measures based on the completion time. The operations of jobs can be nonresumable, resumable or semi-resumable (in general case). It is assumed that setup time is sequence-independent. Their algorithm can be applied to more simple models, too. This problem is general and strongly NP-hard.

3.2. Stochastic Case

In stochastic models, the processing times, the release dates, the starting time and the duration of the unavailability period are not known beforetime; but, it is assumed that the distributions of the processing times, due dates (deadlines), repair time and time at which breakdown occurs are known at time 0. The processing time of a job becomes known only when the J_j is completed. The uptime (A_k) and downtime (B_k) of machines and the processing requirement of the job j (X_j) are assumed to be independent identically distributed random variables.

We consider two sub-cases for the stochastic case: non-resumable and resumable problems. While distribution functions of the uptimes, downtimes and repair time of machines, jobs and the processing times are often the same before, during, and after the interruption(s), but other distribution functions may be different from the original period for the case of some special types of breakdowns.

In general, the breakdown process introduces serious complications. In the literature, single machine problems are often encountered. The performance measures used in the literature are the *expected values of* the *weighted sum of completion times*, the *flow time*, the *weighted sum of number of tardy jobs*, the *maximum expected lateness*, the *expected maximum lateness*, the *availability of heterogeneous systems* with *average response time of multi-class tasks*, the *weighted discounted reward*, the *truncated cost*, the *number of tardy jobs under stochastic order*, the *maximum holding cost*, and the "*expected cost*" where the cost function is proportionate to the *completion time* of J_j.

When preemption is allowed, the decision of which part should be processed is given at time t according to the state of the system. When preemption is not allowed, mostly the sequence of jobs won't be changed if a breakdown occurs. Heuristics that are used for problems when machines are continuously available are also used for the problems in which breakdowns occur. In some studies the conditions under which optimal strategies exist are provided.

3.2.1. Non-resumable availability constraints

Adiri et al. (1989) studied the single machine problem with nonresumable availability constraint in which the machine is subject to breakdowns. The objective was the minimization of the expected

flow time. If the breakdown distribution function over the time is concave, then SPT will asymptotically minimize the expected flow time. For the case of multiple breakdowns, SPT minimizes the expected flow time when the breakdown times are exponentially distributed.

Birge et al. (1990) studied the single machine problem with multiple breakdowns. They considered a simple recourse in their analytical models, in which a permutation schedule - that is fixed a priori - is always maintained and also, certain completion times may be pushed back as a result of one or more breakdowns. The simple recourse has been studied with the objective of minimizing $E[\sum f_j(C_j)]$, where $f_j(t)$ is a nondecreasing real-valued cost function of time t. For the case where we intend to minimize $E[\sum w_jC_j]$, a strong bound on the difference between the optimal policy and the SWPT policy has been provided. Li and Cao (1995) studied a more generalized version of the problem. The single machine is subject to several types of breakdowns according to different probabilities. After the machine breakdown occurs, the jobs' processing times, uptime and the repair time of the machine might be different from the original period. There have been a number of attempts to arrive at the optimal nonpreemptive policies that minimize $E[\sum w_jC_j]$, $E[\sum w_jU_j]$ with constant due dates, and $E[\sum w_jC_j]$ with random due dates. For the single-machine problem, Lee and Lin (2001) assumed that the unavailable time is unknown but with a probabilistic distribution. They have studied the rate-modifying maintenance problems with objective functions such as: expected makespan, total expected completion time, maximum expected lateness, and expected maximum lateness.

A dominance relation for minimizing the makespan with probability 1 was established by Allahverdi (1995) for two-machine flowshop scheduling problem with set-up times and random machine breakdowns. Furthermore, it has been shown that Yoshida and Hitomi's algorithm (1979), YHA, which solves the deterministic problem PF2| S_{ij} |C_{max} optimally in $O(n.logn)$ time, stochastically minimizes the makespan when random breakdowns are present. Allahverdi (1997) published another paper in which he considered the removal times into this latter problem. Allahverdi and Mittenthal (1998) investigated two-machine flow shop scheduling problem with dual criteria (expected makespan and mean flow time) subject to random breakdown.

3.2.2. Resumable availability constraints

Glazebrook (1987) studied the problem 1| *prmp, r*s |$E[\sum f(C_j)]$, where $f(t)$ is a *linear or discounted cost function* of the time t. For the case of geometric uptimes, conditions were given under which breakdowns have no effects on optimal allocation strategies. Two different procedures were given which yield an upper bound on the loss incurred when a processing strategy is adopted under the assumption of no breakdowns or when breakdowns occur in fact. Birge et al. (1990) studied the simple recourse with the multiple breakdowns and the objective of minimizing $E[\sum f_j(C_j)]$, where $f_j(t)$ is a non-decreasing real-valued cost function of time t. Li and Cao (1995) studied a more generalized version of the problem as mentioned in the section 3.2.1. Pinedo (2001) provided the results for the problem 1| *prmp, r*s |$E[\sum w_jC_j]$. He has shown that shortest weighted expected processing time (SWEPT) algorithm solves the single machine problem with multiple breakdowns in order to minimize $E[\sum w_jC_j]$ when uptimes are independent exponential random variables and downtimes are independently and identically distributed geometric random variables. Birge and Glazebrook (1988) studied the single machine problem (1| *prmp, rs* |$E[\sum w_jC_j]$) with multiple breakdowns, too. The state of the system at time t is determined by machine's condition (up or down), the time that machine has been in that condition and the set of completed jobs cumulative processing time up to time t for all unfinished jobs. The decision is made at time t to determine which uncompleted job to process during $[t, t + 1)$. The objective is to minimize the expected weighted flow time ($E[\sum w_jC_j]$). The error and the relative error bounds were provided for the

algorithm which selects the uncompleted job with the largest *Gittins*' index to be processed. In addition to the nonresumable case, Lee and Lin (2001) studied the rate-modifying maintenance single machine scheduling problems for resumable case with the same assumptions and performance measures.

For minimization of $E[C_{max}]$, Schopf and Berman (1999) defined a stochastic scheduling policy based on time-balancing for data parallel applications whose execution behavior can be represented as a normal distribution.

Cai et al. (2005) studied the problem of finding a dynamically optimal policy to process *n* jobs on a single machine subject to stochastic preemptive-repeat-breakdowns (nonresumable operations). Their study allows: 1) the uptimes and downtimes of machines to follow general probability distributions, not necessarily independent of each other; 2) the breakdown process to depend upon the job being processed; and 3) the processing times of jobs to be random variables following arbitrary distributions. They considered two possible cases for the processing time of a job interrupted by a breakdown: a) it is resampled according to its probability distribution or b) it is the same random variable as that before the breakdown. For the problem with resampled processing times, it has been deduced the optimal dynamic policies for criteria including: weighted discounted reward, weighted flowtime, truncated cost, number of tardy jobs under stochastic order, and maximum holding cost. For the problem with the same random processing time, a set of Gittins indices were derived that give the optimal dynamic policies under the criteria of the weighted discounted reward and the weighted flowtime.

Xie and Qin (2006) proposed an algorithm for stochastic scheduling with availability constraints in "*heterogeneous cluster*" to improve the availability of heterogeneous systems while reducing average response time of multi-class tasks. A heterogeneous cluster consists of an array of diverse computers, called computing nodes, which are connected by a high-performance network. To date heterogeneous clusters have been emerging as popular computing platforms for computationally intensive applications with diverse computing needs. Processors operate at different speeds and are not continuously available for processing, in heterogeneous clusters. Indeed, heterogeneous cluster scheduling is a parallel system scheduling problem with high performance object. Examples of such constraints can be found in many areas. For instance, computational nodes in heterogeneous clusters need to be maintained periodically to prevent malfunctions (Lau and Zhang 2004). The "*queue systems*" can be considered in this area, as well.

4. CONCLUSION

Machine scheduling with availability constraint becomes increasingly more important as a better understanding of their importance in various applications is formed. The cause of the machine unavailability might be either deterministic or stochastic. In this paper, the related problems characteristics and literature were presented for both deterministic and stochastic cases with any famous performance metric, job (or operation) resumability and holes cross-ability in the single machine, Pm, PF, F, J, O, FF and FJ environments. This paper can be a good reference for those who are interested in doing research about the *jobs sequencing and scheduling problems under limited resource availability.*

We have summarized the known polynomial (P) and pseudo-P models in Table 1. It is clear that the results of Table 1 are applicable to simpler problems with equivalent performance measures, as well.

Table 1- P and pseudo-P summary (the complexity is stable with the conditions within parentheses).

Model	P and pseudo-P criteria
1,NC| *pmtn* |	$\sum C_j$, C_{max}, L_{max}, $\sum U_j$
1,NC| *pmtn*(M_1), $w_j = t_j$ |	$\sum w_j C_j$
I2,NC| *prec*, $t_j = 1$ |	C_{max}, L_{max}
I2,NC| *pmtn*, *prec* |	C_{max}, L_{max}
I2,NC| *pmtn*, r_j |	$\sum w_j C_j$
Q2,NC| *pmtn* |	$\sum w_j U_j$
I,NC_{zz}| *tree*, $t_j = 1$	C_{max}, L_{max} (in tree)
I,NC_{zz}| *pmtn*, *tree*	C_{max}, L_{max} (in tree)
I,NC_{sc}| *chains*, *pmtn* |	C_{max}
I,NC|	$\sum C_j$ (different beginning times)
I,NC| $t_j = 1$	C_{max} (interval order or chains), L_{max} (in-tree)
I,NC| *pmtn*	C_{max} (chains), L_{max} (eligibility)
I,NC| *pmtn*, r_j |	C_{max}, L_{max}
F2,NC| *ppmtn*(M_1) |	C_{max}
F2,NC| *ppmtn*(M^k_i), $s_1 = s_2$, $e_1 = e_2$ |	C_{max}
F2,NC| *no-wait*(M_i) |	$\sum C_j$
J,NC| $n = 2$	C_{max} (*pmtn*)
O2,NC| *pmtn*(M^k_1 or M^k_2) |	C_{max}
O2,NC| *pmtn*(M_i) |	C_{max}
PF2,NC| *pmtn* (M_1), S_{ij} |	C_{max}

Most of the heuristics with error bound analysis have been gathered in this article. Some of the classical algorithms (for unlimited availability, such as: SPT, SWPT, SWEPT, LRP, JA) are used for the same limited availability models. In some cases those heuristics produce optimal solution(s).

We have summarized the known polynomial (P) and pseudo-P models in Table 1. It is clear that the results of Table 1 are applicable to more simple problems and equivalent performance measures, as well. Most of the problems are NP-hard. Of course, majority of the problems of the FJ environment are strongly NP-hard. Some optimal procedures are provided for problems in class P. The DP approaches are intended for solving the NP-hard problems.

Günter Schmidt has been likely the *most active* author in this filed and McNaughton the first.

If availability constraints come from unexpected breakdowns, fully online algorithms will be needed; but in case of preemptive scheduling, many results of optimality concern the best nearly on-line algorithms. It is an open question to look for the optimality results from fully on-line algorithms and specific availability patterns, or at least to compute performance bounds.

The recent research efforts and papers are considering penalties after the interruption of an operation to enable one to investigate scheduling problems with some setup constraints (namely semi-resumable job sequencing with some extensions). A direction for authors is to assume that one operation cannot be interrupted on any time, but only at given instants. Furthermore, we suggest that authors work on more complicated problems such as sequencing *n* jobs on *m* resources in the FJ or O environment.

REFERENCES

[1] Adiri I., Bruno J., Frostig E., Rinnooy Kan A.H.G. (1989), Single machine flow-time scheduling with a single breakdown; *Acta Informatica* 26; 679-696.

[2] Aggoune R. (2002), Ordonnancement d'Ateliers sous Contraintes de Disponibilité des Machines; *Ph.D. Thesis*, Universite de Metz; France.

[3] Aggoune R. (2004a), Two-Job Shop Scheduling Problems with Availability Constraints. In: *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004)*, Whistler (Canada).

[4] Aggoune R. (2004b), Minimizing the makespan for the flow shop scheduling problem with availability constraints; *European Journal of Operational Research* 153; 534-543.

[5] Aggoune R., Portmann M.-C. (2006), Flow shop scheduling problem with limited machine availability: A heuristic approach; *International Journal of Production Economics* 99; 4-15.

[6] Albers S., Schmidt G. (1999), Scheduling with Unexpected Machine Breakdowns. In: *Computer Science; Vol. 1671, Proceedings of the Third International Workshop on Approximation Algorithms for Combinatorial Optimization Problems: Randomization, Approximation, and Combinatorial Algorithms and Techniques*, Springer-Verlag: London, 269-280.

[7] Albers S., Schmidt G. (2001), Scheduling with unexpected machine breakdowns; *Discrete Applied Mathematics* 110(2-3); 85-99.

[8] Albers S., Schmidt G. (2004), Scheduling with Unexpected Machine Breakdowns; In: Randomization, Approximation, and Combinatorial Optimization (Algorithms and Techniques), Springer: Berlin/ Heidelberg, Volume 1671; 269-280.

[9] Allahverdi A. (1995), Two-stage Production Scheduling with Separated Set-up Times and Stochastic Breakdowns; *Journal of the Operational Research Society* 46(7); 896-904.

[10] Allahverdi A. (1996), Two-machine proportionate flowshop scheduling with breakdowns to minimize maximum lateness; *Computers & Operations Research* 23; 909-916.

[11] Allahverdi A. (1997), Scheduling in stochastic flowshops with independent setup, processing and removal times; *Computers and Operations Research* 24(10); 955-960.

[12] Allahverdi A., Mittenthal J. (1998), Dual criteria scheduling on a two-machine flowshop subject to random breakdowns; *International Transactions in Operational Research* 5; 317-324.

[13] Allaoui H., Artiba A. (2006), Scheduling two-stage hybrid flow shop with availability constraints; *Computers and Operations Research* 33(5); 1399-1419.

[14] Apon A., Wilbur L. (2003), AmpNet - a highly available cluster interconnection network; *Proceedings IEEE International Symposium on Parallel and Distributed Processing*.

[15] Balas E., Lancia G., Serafini P., Vazacopoulos A. (1998), Job shop scheduling with deadlines; *Journal of Combinatorial Optimization* 1(4); 329-353.

[16] Birge J., Frenk J.B.G., Mittenthal J., Rinnooy Kan A.H.G. (1990), Single machine scheduling subject to stochastic breakdowns; *Naval Research Logistics* 37; 661-677.

[17] Birge J., Glazebrook K.D. (1988), Assessing the effects of machine breakdowns in stochastic scheduling; *Operations Research Letters* 7(6); 267- 271.

[18] Blazewicz J., Breit J., Formanowicz P., Kubiak W., Schmidt G. (2001), Heuristic algorithms for the two-machine flowshop problem with limited machine availability; *Omega Journal* 29; 599-608.

[19] Blazewicz J., Dell'Olmo P., Drozdowski M., Maczka P. (2003), Scheduling multiprocessor tasks on parallel processors with limited availability; *European Journal of Operational Research* 149; 377-389.

[20] Blażewicz J., Drozdowski M., Formanowicz P., Kubiak W., Schmidt G. (2000), Scheduling preemtable tasks on parallel processors with limited availability; *Parallel Computing* 26(9); 1195-1211.

[21] Blażewicz J., Ecker K., Pesch E., Schmidt G., Węglarz J. (1996), Scheduling Computer and Manufacturing Processes; Springer; Berlin.

[22] Braun O., Lai T.-C., Schmidt G., Sotskov Y.N. (2002), Stability of Johnson's schedule with respect to limited machine availability; *International Journal of Production Research* 40(17); 4381-4400.

[23] Breit J. (2004), An improved approximation algorithm for two-machine flow shop scheduling with an availability constraint; *Information Processing Letters* 90 (6); 273-278.

[24] Breit J., Schmidt G., Strusevich V.A. (2001a), Two-machine open shop scheduling with an availability constraint; *Operations Research Letters* 29(2); 65-77.

[25] Brucker P., Garey M.R., Johnson D.S. (1977), Scheduling equal-length tasks under treelike precedence constraints to minimize maximum lateness; *Mathematics of Operations Research* 2; 275-284.

[26] Cai X., Wu X., Zhou X. (2005), Dynamically optimal policies for stochastic scheduling subject to preemptive-repeat machine breakdowns; *IEEE Transactions on Automation Science and Engineering* 2(2); 158-172.

[27] Canon C., Billaut J.-C., Bouquard J.-L. (2003), The one-machine sequencing problem with availability constraints; Technical Report 271, Laboratoire d'Informatique de Universitè de Tours; Tours (France).

[28] Carlier J. (1982), The one-machine sequencing problem; *European Journal of Operational Research* 11; 42-47.

[29] Chan F.T.S., Wong T.C., Chan L.Y. (2006), Flexible job-shop scheduling problem under resource constraints; *International Journal of Production Research* 44(11); 2071-2089.

[30] Chen W.J. (2007), Scheduling of jobs and maintenance in a textile company; *International Journal of Advanced Manufacturing Technology* 31; 737-742.

[31] Cheng T.C.E., Liu Z. (2003a), Approximability of two-machine no-wait flowshop scheduling with availability constraints; *Operations Research Letters* 31; 319-322.

[32] Cheng T.C.E., Liu Z. (2003b), 3/2-approximation for two-machine no-wait flowshop scheduling with availability constraints; *Information Processing Letters* 88; 161-165.

[33] Cheng T.C.E., Wang G. (1999), Two-machine flowshop scheduling with consecutive availability constraints; *Information Processing Letters* 71(2); 49-54.

[34] Cheng T.C.E., Wang G. (2000), An improved heuristic for two-machine flowshop scheduling with an availability constraint; *Operations Research Letters* 26; 223-229.

[35] Dehnar Saidy H.R., Taghavi-Fard M.T. (2008), Flexible Job Shop Scheduling Under Availability Constraints; *Journal of Industrial Engineering International*; In press.

[36] Dolev D., Warmuth M.K. (1985a), Scheduling flat graphs; *SIAM Journal on Computing* 14; 638-657.

[37] Dolev D., Warmuth M.K. (1985b), Profile scheduling of opposing forests and level orders. *SIAM Journal on Algebraic and Discrete Methods* 6; 665-687.

[38] Espinouse M.L., Formanowicz P., Penz B. (1999), Minimizing the makespan in the two-machine no-wait flow-shop with limited machine availability; *Computer and Industrial Engineering* 37; 497-500.

[39] Garey M.R., Johnson D.S. (1977), Two-processor scheduling with start-times and deadlines, *SIAM Journal on Computing* 6; 416-426.

[40] Gawiejnowicz S. (2007), Scheduling deteriorating jobs subject to job or machine availability constraints; *European Journal of Operational Research* 180; 472–478.

[41] Glazebrook K.D. (1987), Evaluating the effects of machine breakdowns in stochastic scheduling problems; *Naval Research Logistics* 34; 319-335.

[42] Graham R.L. (1969), Bounds on multiprocessing timing anomalies; *SIAM Journal on Applied Mathematics* 17; 263-269.

[43] Johnson S.M. (1954), Optimal Two and Three Stage Production Schedules with Setup Times Included; *Naval Research Logistics Quarterly* 1(1); 61- 68.

[44] Jungwattanakit J., Reodecha M., Chaowalitwongse P., Werner F. (2008), Algorithms for Flexible Flow Shop Problems with Unrelated Parallel Machines, Setup Times, and Dual Criteria; *International Journal of Advanced Manufacturing Technology* (in press); DOI 10.1007/s00170-007-0977-0.

[45] Kaspi M., Montreuil B. (1988), On the scheduling of identical parallel processes with arbitrary initial processor available time; Research Report, School of Industrial Engineering; Purdue University.

[46] Kellerer H. (1998), Algorithms for multiprocessor scheduling with machine release time; *IIE Transactions* 30(11); 991-999.

[47] Kubiak W., Blażewicz J., Formanowicz P., Breit J., Schmidt G. (2002), Two-machine flow shops with limited machine availability; *European Journal of Operational Research* 136(3); 528-540.

[48] Kubiak W., Blażewicz J., Formanowicz P., Schmidt G. (1997), A branch and bound algorithm for two machine flow shop with limited machine availability; *The Abstracts of the Tenth Meeting of the European Chapter on Combinatorial Optimization* 38.

[49] Kubzin M.A., Strusevich V.A., Breit J., Schmidt G. (2005), Polynomial-time approximation schemes for two-machine open shop scheduling with nonavailability constraints; *Naval Research Logistics* 53(1); 16-23.

[50] Lau H. C., Zhang C. (2004), Job Scheduling with Unfixed Availability Constraints; *Proceeding of 35th Meeting of the Decision Sciences Institute* (DSI), Boston (USA), 4401-4406.

[51] Lawler E.L., Martel C.U. (1989), Preemptive scheduling of two uniform machines to minimize the number of late jobs; *Operations Research* 37; 314-318.

[52] Leangsuksun C., Tikotekar A., Pourzandi M., Haddad I. (2005), Feasibility study and early experimental results towards cluster survivability; *Proceedings of the IEEE International Symposium on Cluster Computing and the Grid*, 77-81.

[53] Lee C. Y. (1991), Parallel machine scheduling with nonsimultaneous machine available time; *Discrete Applied Mathematics* 30; 53-61.

[54] Lee C. Y. (1996), Machine scheduling with an availability constraint; *Journal of Global Optimization* 9; 395-416.

[55] Lee C. Y. (1997), Minimizing the makespan in two-machine flowshop scheduling problem with an availability constraint; *Operations Research Letters* 20; 129-139.

[56] Lee C. Y. (1999), Two-machine flowshop scheduling with availability constraints; *European Journal of Operational Research* 114; 420-429.

[57] Lee C. Y., Lei L., Pinedo M. (1997), Current trends in deterministic scheduling; *Annals of operations Research* 70; 1-41.

[58] Lee C. Y., Liman S.D. (1992), Single machine flow-time scheduling with scheduled maintenance; *Acta Informatica* 29; 375-382.

[59] Lee C. Y., Liman S.D. (1993), Capacitated two-parallel machines scheduling to minimize sum of job completion times; *Discrete Applied Mathematics* 41; 211-222.

[60] Lee C. Y., Lin C.S. (2001), Single-machine scheduling with maintenance and repair rate-modifying activities; *European Journal of Operational Research* 135; 493-513.

[61] Leon V. J., Wu S. D. (1992), On scheduling with ready-times, due-dates and vacations; *Naval Research Logistics* 39; 53-65.

[62] Levitin G. (2000), Multistate Series-Parallel System Expansion-Scheduling Subject to Availability Constraints; *IEEE Transactions on Reliability* 49(1); 71-79.

[63] Li W., Cao J. (1995), Stochastic scheduling on a single machine subject to multiple breakdowns according to different probabilities; *Operations Research Letters* 18; 81-91.

[64] Lin G.H., Yao E.Y., He Y. (1998), Parallel machine scheduling to maximize the minimum load with nonsimultaneous machine available times; *Operations Research Letters* 22; 75-81.

[65] Liu Z., Sanlaville E. (1995a), Preemptive scheduling with variable profile, precedence constraints and due dates; *Discrete Applied Mathematics* 58; 253-280.

[66] Liu Z., Sanlaville E. (1995b), Profile scheduling of list algorithms. In: Chretienne, P. et al. (Ed), Scheduling Theory and its Applications, John Wiley & Sons: New York, 91-110.

[67] Lorigeon T., Billaut J. C., Bouquard J. L. (2002), A dynamic programming algorithm for scheduling jobs in a two-machine open shop with an availability constraint; *Journal of the Operational Research Society* 53 (11); 1239-124.

[68] Mauguière P., Billaut J. C., Bouquard J. L. (2003a), Scheduling resumable and non-resumable operations; In: *Proceedings of the Joint International Meeting EURO/INFORMS*, Istanbul (Turkey), 166-167.

[69] Mauguière P., Bouquard J. L., Billaut J. C. (2003b), A branch and bound algorithm for a job shop scheduling problem with availability constraints; In: *Proceedings of the Sixth Workshop on Models and Algorithms for Planning and Scheduling Problems, MAPSP'2003*, Aussois (France), 147-148.

[70] Mauguière P., Billaut J. C., Bouquard J. L. (2005), New single machine and job shop scheduling problems with availability constraints; *Journal of Scheduling* 8; 211-231.

[71] McNaughton R. (1959), Scheduling with deadlines and loss functions, *Management Science* 6; 1-12.

[72] Moore J.M. (1968), An *n* job, one machine sequencing algorithm for minimizing the number of late jobs; *Management Science* 15; 102-109.

[73] Mosheiov G. (1994), Minimizing the sum of job completion times on capacitated parallel machines; *Mathematical and Computer Modelling* 20; 91-99.

[74] Ng C.T., Kovalyov M.Y. (2003), An FPTAS for scheduling a two-machine flowshop with one unavailability interval; *Naval Research Logistics* 51(3); 307-315.

[75] Papadimitriou C.H., Yanakakis M. (1979), Scheduling interval ordered tasks; *SIAM Journal on Computing* 8; 405-409.

[76] Pinedo M. (2001), Scheduling: Theory, Algorithms and Systems; 2nd edition, Prentice-Hall; Englewood Cliffs, NJ.

[77] Sanlaville E. (1995), Nearly on line scheduling of preemptive independent tasks; *Discrete Applied Mathematics* 57; 229-241.

[78] Sanlaville E., Schmidt G. (1998), Machine scheduling with availability constraints; *Acta Informatica* 35; 795-811.

[79] Schmidt G. (1984), Scheduling on semi-identical processors; *Zeitschrift für Operations Research* A28; 153-162.

[80] Schmidt G. (1988), Scheduling independent tasks with deadlines on semi-identical processors; *Journal of Operational Research Society* 39; 271-277.

[81] Schmidt G. (2000a), Performance guarantee of two simple priority rules for production scheduling; *International Journal of Production Economics* 68(2); 151-159.

[82] Schmidt G. (2000b), Scheduling with limited machine availability; *European Journal of Operational Research* 121; 1-15.

[83] Schopf J.M., Berman F. (1999), Stochastic Scheduling; *Proceedings of the ACM/IEEE Conference Supercomputing*. Portland-Oregon (United States of America).

[84] Sheen G. J., Liao L. W. (2007), Scheduling machine-dependent jobs to minimize lateness on machines with identical speed under availability constraints; *Computers & Operations Research* 34(8); 2266-2278.

[85] Ullman J.D. (1975), NP-complete scheduling problems; *Journal of Computer and System Sciences* 10; 384-393.

[86] Wang G., Cheng T.C.E. (2001), Heuristics for two-machine no-wait flowshop scheduling with an availability constraint; *Information Processing Letters* 80; 305-309.

[87] Wang X., Cheng T.C.E. (2006), Machine scheduling with an availability constraint and job delivery coordination; *Naval Research Logistics* 54(1); 11-20.

[88] Wang X., Cheng T.C.E. (2007a), Heuristics for two-machine flowshop scheduling with setup times and an availability constraint; *Computers & Operations Research* 34; 152-162.

[89] Wang X., Cheng T.C.E. (2007b), An approximation scheme for two-machine flowshop scheduling with setup times and an availability constraint; *Computers and Operations Research* 34(10); 2894-2901.

[90] Wu C.C., Lee W.C. (2003), Scheduling linear deteriorating jobs to minimize makespan with an availability constraint on a single machine; *Information Processing Letters* 87 (2); 89–93.

[91] Xie T., Qin X. (2006), Stochastic Scheduling with Availability Constraints in Heterogeneous Clusters; *Proceedings of the 8th IEEE International Conference on Cluster Computing* (*Cluster 2006 IEEE*), 1-10.

[92] Yoshida T., Hitomi K. (1979), Optimal two-stage production scheduling with setup times separated; *AIIE Transactions* 11; 261–263.

[93] Breit J., Schmidt G., Strusevich V.A., "Non-preemptive two-machine open shop scheduling with non-availability constraints", Selected Research Papers/ Abstracts, Source: *Mathematical Methods of Operation Research* 57; http://www.itm.uni-sb.de/staff/gs/abstracts.htm, June 2001b.

[94] Türkcan A., "Machine Scheduling with Availability Constraints", http://citeseer.ist.psu.edu/turkcan99machine.html, April 15 1999.