Shiraz Journal of System Management
Vol 2, No. 4, Ser. 8 (2014), 001-016

# A Heuristic Algorithm for Solving Single Machine Scheduling Problem with Periodic Maintenance

**Amir Ebrahimi Zade**
Department of Industrial Engineering,
Yazd University, Yazd, Iran
(Corresponding Author)

**Mohammad Bagher Fakhrzad**
Department of Industrial Engineering,
Yazd University, Yazd, Iran

**Mohsen Hasaninezhad**
Department of Industrial Engineering,
Iran University of Science and Technology, Tehran, Iran

**Abstract.** In this paper, the scheduling problem with nonresumable jobs and maintenance process is considered in order to minimize the makespan under two alternative strategies. The first strategy is to implement the maintenance process on the machine after a predetermined time period and the second one is to consider the maximum number of jobs that can be done with an especial tool. We propose a new mathematical formulation for the aforementioned problem and regarding the problem is included in the NP-Hard class of problems, in the second part of the paper, we propose a heuristic algorithm in order to solve the problem in a reasonable time. Also we compare the performance of the proposed algorithm with existing methods in the literature. Computational results showed that the proposed algorithm is able to attain optimum solutions in most cases

and also corroborate its better performance than the existing heuristic methods in the literature.

**Keywords:** Single Machine Scheduling, Periodic Maintenance, Nonresumable Jobs, Mathematical Formulation, Heuristic Algorithm.

## 1. Introduction and Literature Review

In the most of the scheduling problems, it is assumed the machine is continuously and uninterruptedly available. However in the real world problems this is not the case [1]. The possible machine breakdown and interruption would be unavoidable in case of working continuously and also this may affect the quality of the jobs being processed by the machine. Thus, the periodic preventive maintenance process is usually planned and implemented to avoid the aforementioned problem. In this paper the scheduling problem is mathematically formulated subject to the amount of time the machine is available between the two consecutive periods, at the end of which the maintenance process is implemented and the maximum number of jobs which could be processed by the machine over the operating time, thereafter an efficient heuristic is proposed to solve the problem in a reasonable computational time.

Jobs are done in two different ways when there is a limited time period in which the machine is available before the next maintenance process. In first case, which is called preemptive jobs, it is assumed that operations of the in-process jobs could be continued after machine interruption. On the other hand for non-preemptive jobs, it is assumed that the incomplete jobs must be re-processed after interruption. Also, for some manufacturing processes such as manufacturing of the electronic circuits, the number of jobs to process before changing the tools is a constraint. Considering the Graham's three–field notation for symbolizing the scheduling problems, this case is specified with 1/nr-pm/cmax [2].

Problem of machine availability has been widely considered in the literature due to machine breakdown, tool changing and preventive maintenance process. In some of the researches, machine unavailability has been considered due to the stochastic breakdowns. For single machine scheduling problem with stochastic breakdowns, the optimal

solution has been provided in [3-5]. Some heuristics have been also developed to minimize the number of tardy jobs or makespan [6, 7]. In addition, the constraint of the machine unavailability due to tool change has been also considered in [8-12]. Machine unavailability due to preventive maintenance process has been considered based on both flexible and periodic maintenance process. In the flexible maintenance process, the earliest and latest start time of the maintenance process is already specified and the machine shut down is allowed in this range. Yang et al. surveyed the problem of the single machine scheduling considering the only one flexible maintenance process. Regarding the actual time needed for maintenance process is shorter than or equal to predetermined time, the flexible maintenance process is considered. They surveyed the problem to minimize the makespan and prove the NP-Hardness of the developed problem [13]. Also Qi and chen in [14] considered the problem of scheduling several processes of maintenance as well as jobs processing on the machine. They applied heuristics and also methods on basis of the reputable branch and bound to determine the best jobs sequence and scheduling of the maintenance process in order to minimize the makespan. A mixed binary integer programming and a heuristic to minimize the number of in-process jobs have been also proposed in [15]. Finally, Low et al. in [16] surveyed the problem of single machine scheduling considering two alternative strategies, namely, machine unavailability after a fixed time period and also after processing a specific number of jobs to change the tool. They considered minimization of the makespan as their objective.

In context of the problems with periodic maintenance process, there are variants of research. Liao and Chen developed an algorithm on basis of the reputable branch and bound to minimize the maximum tardiness [17]. Chen suggested a heuristic algorithm to minimize the makespan [18]. In addition, [19] presented a method based on the reputable branch and bound as well as a heuristic to minimize the number of tardy jobs. Lee and Lee developed a heuristic to minimize the makespan [20]. Lau and Zhang assumed that a fixed number of operations must be done between two successive interruptions [21]. Articles [22-24] assumed that the machine must stop working utmost after processing a fixed number of jobs due to the maintenance process. Finally in [25], Hsu et al

considered the single machine scheduling problem with periodic maintenance and the same strategies as already considered ones in [16], their objective was to minimize Cmax. They developed a two-level integer programming for solving the problem optimally as well as two heuristics for solving the large scale problems, namely, DBF and BBF. As Pinedo mentioned, the scheduling problems belong to strongly NP-Hard problems [26]. Also In the single machine scheduling problems with machine unavailability, there are a lot of papers in which NP-Hardness of the problem is mentioned for example [27] proved the NP-Hardness of the problems with deterministic maintenance times. In this paper, for solving the single machine scheduling problem with fixed time between two consecutive maintenance operations and maximum number of jobs that can be done during this period due to change the tool, a heuristic algorithm is proposed.

The remaining of this paper is organized as follows: In the following section, we first propose a new mathematical model for solving the problem and following that, an efficient heuristic algorithm is proposed to solve the problem in a reasonable computational time. At the third part, the results of applying the proposed methods on some experimental problems are presented and compared against each other. Conclusions and some guidelines for future study are presented in last part of this paper.

## 2. Proposed Methods

This section contains two parts in each of which a separate solution method to solve the aforementioned problem is provided. In the first part, a mixed integer programming model is proposed for the first time in the literature, whereas in the second part an efficient heuristic algorithm is proposed to solve the problem.

### 2.1. Mathematical Formulation

In this paper, a single machine scheduling problem is considered with two alternative strategies: 1- implementing the maintenance process after a predetermined time period and 2- maximum number of jobs that can be done during this period in order to change the tool. The objective of

the problem is to minimize the makespan and it is assumed that jobs are nonresumable which means if process of any job couldn't complete till the next maintenance process, it must be repeated after maintenance time. Considering all above, we can define the problem as bellow:

$n$ jobs are planned to be processed on a machine and are nonresumable. The processing time of each job is indicated with $p_i$ and it is assumed that all jobs are ready and available at beginning. Time period between two successive maintenance processes is $t$ and time needed for implementing any maintenance activity is specified with $m$. The jobs processed between two successive maintenance processes are called a batch and should not exceed $k$. Hence, if it is supposed to have $l$ batches in a time horizon, the number of maintenance processes is $l$-1. This could be shown in the schematic view in Figure 1.

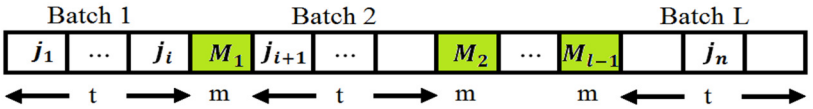


**Figure 1.** A schematic view of the problem

Considering the above-mentioned explanation, the problem notation is as follows:

| | |
|---|---|
| $t$ | Time period between two successive maintenance processes |
| $m$ | Time needed for implementing any maintenance activity |
| $k$ | Maximum number of jobs to process in period $t$ |
| $M$ | A large number |
| $n$ | Number of jobs |
| $p_i$ | Processing time of job $i$; $i = 1, 2, \ldots, n$ |
| $x_{ij}$ | If job $i$ belongs to batch $j$, it equals 1, Otherwise it equals 0; $j = 1, 2, \ldots, l$ |
| $y_j$ | If at least one job belongs to batch $j$, it equals 1, otherwise this variable equals 0. |
| $g_j$ | Amount of gap for batch $j$ |
| $w_j$ | The amount of gap for all batches except for empty batches and the last batch in which jobs take place |

In [25] the problem of minimizing the makespan is solved by using two mathematical models, in the first of which the minimum number of required batches to complete jobs are determined and in the second of which, assuming the number of batches a priori, the jobs are sequenced and arranged in the batches such that the makespan is minimized. In this paper, a new mathematical formulation is proposed which solves the problem in one stage.

Before representing the mathematical model, the general idea of solving the problem is to be explained. Regarding the optimal number of the batches is not specified at the beginning, the maximum possible number of batches is considered initially. Hence, if there are $n$ jobs, there could be $n$ batches at most. However, after arranging the jobs considering minimization of the makespan, subject to predetermined maximum time period allowed and the maximum number of jobs in the batches, there would be batches with more than one job assigned and, as a result, there are some batches with no jobs assigned. The model omits these empty batches. Also the amount of gap for each batch, excluding the last batch, is calculated. At last, the sum of the jobs' processing times, gaps of the batches and fixed times needed for maintenance processes constitute the amount of the makespan.

Following the explanation above, the mixed integer programming model is proposed as follows:

$$\min \left[\left(\sum_j y_j\right) - 1\right] \cdot m + \sum_j w_j + \sum_i p_i \tag{1}$$

$$\sum_j x_{ij} = 1 \qquad \forall\, i \tag{2}$$

$$\sum_i x_{ij} \le k \qquad \forall\, j \tag{3}$$

$$\sum_i x_{ij} p_i \le t \qquad \forall\, j \tag{4}$$

$$\sum_i x_{ij} \le M \cdot y_j \qquad \forall\, j \tag{5}$$

$$\sum_i x_{ij} > M(y_j - 1) \qquad \forall\, j \tag{6}$$

$$y_{j+1} \leq y_j \qquad \forall\, j \tag{7}$$

$$g_j = t - \sum_i x_{ij} p_i \qquad \forall\, j \tag{8}$$

$$w_j = g_j \cdot y_{j+1} \qquad \forall\, j \tag{9}$$

$$y_j, x_{ij} \in \{0,1\},\ g_j,\ w_j \geq 0 \tag{10}$$

Expression (1) is objective function which, as described above, consists of three separate terms. The first term calculates the sum of the fixed times needed for maintenance processes, the second term calculates the sum of the gaps of the batches excluding the last one and the third one calculates the sum of the jobs' processing times. Constraint (2) assures that each job is exactly assigned to one batch. Constraint (3) assures that the number of jobs in each batch does not exceed $k$. Constraint (4) assures that the sum of the jobs' processing times assigned to one batch could not exceed the maximum time period allowed $t$.

Initially $n$ empty batches were considered and this may result in empty batches at last, so (5) and (6) are regulated so that if any job is assigned to the $k^{\text{th}}$ batch, respective binary variable $y$ takes the value 1 and otherwise it takes 0. Constraint (7) is regulated so that until a batch is empty, all of its further batches will also be empty. Hence, this constraint helps the empty batches, which corresponding binary variable $y$ is 0, take place just after the filled batches. This characteristic, as will be explained later in equation (8), is used to calculate the amount of gap. By using (8), the initial value of gap is calculated. If we are going to calculate the amount of gap with (8), we will face a problem. In this situation an empty batch will have gap $t$ and it will affect the objective function, in order to solve this problem equation (9) is introduced to calculate the amount of variable $w$. for empty batches the amount of this variable will equal 0 also because the amount of gap for the last batch in which jobs take place, is not a part of the makespan, the amount of this variable for the mentioned batch will equal 0. This variable for all other batches will be equal to gap for that batch. In order to linearize this constraint we propose below constraints:

$$w_j - My_{j+1} \le g_j, \qquad \forall j \tag{11}$$

$$w_j + M(1 - y_{j+1}) \ge g_j \qquad \forall j \tag{12}$$

In these constraints, if any job is assigned to the next batch, $w_j = g_j$, but if the next batch is empty, upper bound for $w_j$ is $g_j$ and considering it only takes the positive values, the mentioned variable ranges from 0 to $g_j$. On the other hand, considering this variable is a part of the objective function and the minimization of the objective function is the case, so this variable takes its least possible amount, say 0.

## 2.2. Proposed Heuristic Algorithm

As mentioned earlier, the investigated problem belongs to the NP-hard class and solving large scale NP-hard problems requires considerable computational time. Therefore a heuristic algorithm with 8 steps is proposed to solve the problem.

***Step 1:*** Calculate the average processing time of all jobs ($\mu_p$).

***Step 2:*** classify the jobs according to their processing time in to smaller (G1) and larger (G2) than $\mu_p$.

***Step 3:*** Arrange the jobs in decreasing order of processing times in each group.

***Step 4:*** Create a new batch and start allocating jobs with the first unassigned job in G2 (job with the maximum processing time). Continue allocating subsequent jobs until $\sum p_i > t$. otherwise go to step5. Whenever the number of jobs assigned to a batch exceeds $k$, create a new batch and continue allocation.

***Step 5:*** Select the first unassigned job from the end of G2 and allocate it to the current list. Continue allocating further jobs until $\sum p_i > t$. otherwise go to step 6. Whenever the number of jobs assigned to a batch exceeds $k$, create a new batch and resume allocation.

***Step 6:*** Select the first unassigned job from the beginning of G1 and allocate it to the current list. This process holds on until

$\sum p_i > t$, otherwise go to step 7. If the number of assigned jobs exceeds $k$, crate a new list and continue allocation.

***Step 7:*** Select the first job from the end of G1 and allocate it to the current list. Continue allocating subsequent jobs until $\sum p_i > t$. if the batch size exceeds $k$, create a new batch and continue allocation. If there are any jobs left go to step 4, otherwise go to 8.

***Step 8:*** End.

To have a better understanding from the proposed algorithm, a numerical example is provided in Table 1, which is comprised of 10 jobs and associated process times. Suppose $t = 20$, $k = 3$ and $m = 8$. According to the first step, $\mu_p$ equates 10.2.

**Table 1.** Jobs and their processing times

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $p_i$ | 2 | 17 | 4 | 7 | 10 | 12 | 9 | 10 | 17 | 14 |

The next step is to divide jobs into two groups and sort them in descending order of processing times which the results are presented in table 2.

**Table 2.** Dividing and sorting jobs

| G1 | $i$ | 5 | 8 | 7 | 4 | 3 | 1 |
|---|---|---|---|---|---|---|---|
| | $p_i$ | 10 | 10 | 9 | 7 | 4 | 2 |

| G1 | $i$ | 2 | 9 | 10 | 6 |
|---|---|---|---|---|---|
| | $p_i$ | 17 | 17 | 14 | 12 |

Job (2) with maximum processing time ($p_2$=17) is selected from G2 and allocated to the first batch, $B_1$. Thereafter, job 9 is selected from G2 but it cannot be allocated to the first batch, because $\sum p_i$>20. In other words, the next job processing time should be less than or equal to 3 so that it can be allocated to the $B_1$. Accordingly both job 6 nor 5 is allocated to B1 and job 1 will be added to $B_1$. To allocate remaining jobs, another list should be created. Results of algorithm steps repetition for all jobs are summarized in table 3. Finally, after repetition of algorithm steps, all

jobs will be allocated to six batches which means five maintenance operations.

**Table 3.** Allocating jobs based on proposed algorithm

| $B_1$ | | $B_2$ | | $B_3$ | | $B_4$ | | $B_5$ | | $B_6$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $i$ | $P_i$ | $i$ | $P_i$ | $i$ | $P_i$ | $i$ | $P_i$ | $i$ | $P_i$ | $i$ | $P_i$ |
| 2 | 17 | 9 | 17 | 10 | 14 | 6 | 12 | 5 | 10 | 7 | 9 |
| 1 | 2 | | | 3 | 4 | 4 | 7 | 8 | 10 | | |

## 3. Computational Results

To evaluate the performance of the proposed algorithm, sixty different experimental problems were designed. These problems are generated in the following way:

1. The job size $n$ can be 20, 30, 40, 50, 100, 250, 300, 350, 400, and 500.
2. There are 6 different test problems for each job size to be tested.
3. The processing time for a job, $p_i$ and maintenance time for a machine, $t$ are integers uniformly distributed in [1, 5, 10], respectively.
4. The length of the time interval between two consecutive maintenance periods is $m = \max\left\{\left\lceil a\sum_i p_i \right\rceil, \max p_i\right\}$
5. The maximum number of jobs processed in the machine's available time interval is $k = \lceil bn \rceil$ where b= 1, 1/2, 1/3, 1/4, 1/5.

Optimal solution for each problem is obtained by GAMS 22.2 with CPLEX solver and the heuristic algorithm was coded in MATLAB R2011 b.

Results on all 60 experimental problems showed that the proposed algorithm has similar or better performance than BDF algorithm introduced in [25]. Furthermore, as can be seen in table 4, there is no significant difference between proposed algorithm and optimal solutions of the problems which indicates the efficiency of proposed algorithm.

Also the average computational time required by the proposed algorithm and the average computational time required for optimal solutions are

compared in table 5. As evident in this table, the proposed heuristic leads to a considerable reduction in the computational time versus the optimal solution found by GAMS.

**Table 5.** Average computational time

| Size | Optimal Solution | Heuristic Solution |
|---|---|---|
| 20 | 0.441 | 0.035 |
| 30 | 0.824 | 0.022 |
| 40 | 2.545 | 0.078 |
| 50 | 4.925 | 0.090 |
| 100 | 26.260 | 0.064 |
| 250 | 731.325 | 0.005 |
| 300 | 1190.108 | 0.094 |
| 350 | 2663.199 | 0.134 |
| 400 | 7523.377 | 0.058 |

Also table 6 compares the proposed algorithm versus the BDF algorithm due to the average deviation from optimal solutions. The better performance of the proposed algorithm compared to the existing algorithms is quite clear from the table. In all generated problems, the average error of the proposed algorithm is less than the BDF algorithm. The maximum error of BDF algorithm is approximately 0.11% and it occurs on problems with 50 jobs, while the maximum error of the proposed algorithm occurs on problems with 500 jobs and is about 0.02%.

**Table 6.** Average deviation from the optimal solution

| Size | Heuristic %Error | BDF %Error |
|---|---|---|
| 20 | 0.001 | 0.004 |
| 30 | 0.005 | 0.011 |
| 40 | 0.005 | 0.032 |
| 50 | 0.005 | 0.113 |
| 100 | 0.004 | 0.069 |
| 250 | 0.004 | 0.007 |
| 300 | 0.010 | 0.024 |
| 350 | 0.011 | 0.099 |
| 400 | 0.018 | 0.044 |

## 4. Conclusion

In this paper, a single machine scheduling problem with machine interruption due to the periodic maintenance process and constraint on the maximum number of jobs processed in each batch on the machine is considered. A new mixed integer programming formulation is proposed to solve this problem and considering the problem belongs to the NP-Hard class, trying to solve it optimally may cause extremely long computational time. So a heuristic algorithm was proposed that, as computational results shows, can achieve high quality solution (and in most cases the optimal solution) in a considerably shorter time.

For the interested researchers in this field, as the future studies, it is suggested to compare the performance (due to time factor) of the proposed mixed integer model in this paper with the model proposed in [25].

**Table 4.** Comparison between the optimal solutions, proposed heuristic, and existing heuristic namely BDF

| Size | Solution found | | | Error percentage | |
|---|---|---|---|---|---|
| | GAMS | BDF | Proposed heuristic | BDF | Proposed heuristic |
| 20 | 110 | 110 | 110 | 0.000 | 0.000 |
| 20 | 111 | 113 | 111 | 0.018 | 0.000 |
| 20 | 85 | 85 | 85 | 0.000 | 0.000 |
| 20 | 155 | 156 | 156 | 0.006 | 0.006 |
| 20 | 140 | 140 | 140 | 0.000 | 0.000 |
| 20 | 212 | 212 | 212 | 0.000 | 0.000 |
| 30 | 198 | 205 | 203 | 0.034 | 0.025 |
| 30 | 205 | 205 | 205 | 0.000 | 0.000 |
| 30 | 132 | 132 | 132 | 0.000 | 0.000 |
| 30 | 160 | 160 | 160 | 0.000 | 0.000 |
| 30 | 205 | 205 | 205 | 0.000 | 0.000 |
| 30 | 152 | 157 | 157 | 0.032 | 0.032 |
| 40 | 225 | 233 | 233 | 0.034 | 0.034 |
| 40 | 249 | 261 | 260 | 0.046 | 0.042 |
| 40 | 246 | 246 | 246 | 0.000 | 0.000 |
| 40 | 193 | 208 | 208 | 0.072 | 0.072 |
| 40 | 241 | 251 | 251 | 0.040 | 0.040 |
| 40 | 227 | 227 | 227 | 0.000 | 0.000 |
| 50 | 258 | 311 | 275 | 0.170 | 0.066 |

| Size | Solution found | | | Error percentage | |
|---|---|---|---|---|---|
| | GAMS | BDF | Proposed heuristic | BDF | Proposed heuristic |
| 50 | 305 | 410 | 322 | 0.256 | 0.056 |
| 50 | 295 | 314 | 314 | 0.061 | 0.061 |
| 50 | 276 | 331 | 288 | 0.166 | 0.043 |
| 50 | 269 | 272 | 272 | 0.011 | 0.011 |
| 50 | 271 | 275 | 275 | 0.015 | 0.015 |
| 100 | 643 | 776 | 662 | 0.171 | 0.030 |
| 100 | 627 | 643 | 643 | 0.025 | 0.025 |
| 100 | 708 | 861 | 708 | 0.178 | 0.000 |
| 100 | 545 | 545 | 545 | 0.000 | 0.000 |
| 100 | 1607 | 1608 | 1608 | 0.001 | 0.001 |
| 100 | 508 | 529 | 529 | 0.040 | 0.040 |
| 250 | 1832 | 1832 | 1832 | 0.000 | 0.000 |
| 250 | 1400 | 1400 | 1400 | 0.000 | 0.000 |
| 250 | 1331 | 1331 | 1331 | 0.000 | 0.000 |
| 250 | 1310 | 1334 | 1330 | 0.018 | 0.015 |
| 250 | 1415 | 1415 | 1415 | 0.000 | 0.000 |
| 250 | 1229 | 1258 | 1254 | 0.023 | 0.020 |
| 300 | 1451 | 1506 | 1506 | 0.037 | 0.037 |
| 300 | 1618 | 1697 | 1693 | 0.047 | 0.044 |
| 300 | 1627 | 1647 | 1647 | 0.012 | 0.012 |
| 300 | 1573 | 1573 | 1573 | 0.000 | 0.000 |
| 300 | 1663 | 1750 | 1750 | 0.050 | 0.050 |
| 300 | 1591 | 1591 | 1591 | 0.000 | 0.000 |
| 350 | 1984 | 1984 | 1984 | 0.000 | 0.000 |
| 350 | 2009 | 2610 | 2120 | 0.230 | 0.055 |
| 350 | 1921 | 2027 | 2027 | 0.052 | 0.052 |
| 350 | 1875 | 2542 | 1994 | 0.262 | 0.063 |
| 350 | 5110 | 5110 | 5110 | 0.000 | 0.000 |
| 350 | 1938 | 2043 | 2043 | 0.051 | 0.051 |
| 400 | 2117 | 2568 | 2312 | 0.176 | 0.092 |
| 400 | 4048 | 4048 | 4048 | 0.000 | 0.000 |
| 400 | 2166 | 2245 | 2245 | 0.035 | 0.035 |
| 400 | 2142 | 2264 | 2264 | 0.054 | 0.054 |
| 400 | 2024 | 2024 | 2024 | 0.000 | 0.000 |
| 400 | 3210 | 3213 | 3213 | 0.001 | 0.001 |
| 500 | 2901 | 3052 | 3052 | 0.049 | 0.049 |
| 500 | 2680 | 2793 | 2793 | 0.040 | 0.040 |
| 500 | 2657 | 2657 | 2657 | 0.000 | 0.000 |
| 500 | 2782 | 2895 | 2895 | 0.039 | 0.039 |
| 500 | 2552 | 2628 | 2628 | 0.029 | 0.029 |

Furthermore, since this issue has not been studied in case of Meta heuristic algorithms yet, evaluation of meta-heuristic algorithms such as simulated annealing, genetic algorithm, tabu search or particle swarm optimization algorithm is suggested as a future research. Another interesting future study is to consider the jobs due dates and formulating the problem such that the number of tardy jobs is minimized.

## References

[1] Pinedo, M.L. (2002). *Scheduling: Theory, Algorithms, and Systems* (Vol. 1) New Jersey: Springer.

[2] Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. H. G. Rinnooy. (1979). *Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey.* In E. L. J. P.L. Hammer & B. H. Korte (Eds.), *Annals of Discrete Mathematics;* 5: 287-326.

[3] Pinedo, M., Rammouz, E. (1988). *A note on stochastic scheduling on a single machine subject to breakdown and repair.* Probability in the Engineering and Information- National Sciences; 2:41–9.

[4] Birge, J., Frenk, JBG, Mittenthal, J., Rinnooy Kan, AHG. (1990). *Single-machine scheduling subject to stochastic breakdowns.* Naval Research Logistics; 37:664–77.

[5] Frostig, E. (1991). *A note on stochastic scheduling on a single machine subject to breakdown-the preemptive repeat model.* Probability, Engineering and Informational Sciences; 5:349–54.

[6] Adiri, I., Bruno, J., Frostig, E., Rinnooy Kan, AHG. (1989). *Single machine flow-time scheduling with a single breakdown.* Acta Informatica; 26:679–96.

[7] Adiri, I., Frostig, E., Rinnooy Kan, AHG. (1991). *Scheduling on a single machine with a single breakdown to minimize stochastically the number of tardy jobs.* Naval Research Logistics; 38:261–71.

[8] Akturk, MS., Ghosh, JB., Gunes, ED. (2003). *Scheduling with tool changes to minimize total completion time: a study of heuristics and their performance.* Naval Research Logistics; 50:15–30.

[9] Akturk, MS., Ghosh, JB., Gunes ED. (2004). *Scheduling with tool changes to minimize total completion time: basic results and SPT performance*. European Journal of Operations Research; 157:784–90.

[10] Ecker, KH. Gupta, JND. (2005). *Scheduling tasks on a flexible manufacturing machine to minimize tool change delays*. European Journal of Operations Research; 164:627–38.

[11] Chen, JS. (2008). *Optimization models for the tool change scheduling problem*. Omega; 36:888–94.

[12] Choi, Y-C., Kim, Y-D. (2001). *Tool replacement policies for a machining center producing multiple types of products with distinct due dates*. International Journal of Production Research; 39:907–21.

[13] Yang, D.L., Hung, C.L., Hsu, C.J., Chern, M.S., (2002). *Minimizing the makespan in a single machine scheduling problem with a flexible maintenance*. Journal of the Chinese Institute of Industrial Engineers; 19: 63–66.

[14] Qi, X., Chen, T., Tu. F. (1999). *Scheduling the maintenance on a single machine*. Journal of the Operational Research Society; 50:1071–8.

[15] Chen, JS. (2006). *Single machine scheduling with flexible and periodic maintenance*. Journal of the Operational Research Society; 57:703–10.

[16] Low, C., Ji, M., Hsu, C-J., Su, C-T. (2010). *Minimizing the makespan in a single machine scheduling problems with flexible and periodic maintenance*. Applied Mathematical Modeling; 34: 334–42.

[17] Liao, CJ. Chen, WJ. (2003). *Single-machine scheduling with periodic maintenance and nonresumable jobs*. Computers and Operations Research; 30:1335–47.

[18] Chen, WJ. (2006). *Minimizing total flow time in the single machine scheduling problem with periodic maintenance*. Journal of the Operational Research Society; 57:410 –5.

[19] Chen, WJ. (2009). *Minimizing number of tardy jobs on a single machine subject to periodic maintenance*. Omega; 37:591–9.

[20] Lee, S-H., Lee, I-G. (2008). *Heuristic algorithm for the single machine scheduling with periodic maintenance*. Journal of the Korean Institute of Industrial Engineers; 34:318–27.

[21] Lau, HC. Zhang, C. (2004). *Job scheduling with unfixed availability constraints*. In: Proceedings of 35th Meeting of the Decision Sciences Institute. Boston, USA; p. 4401–6.

[22] Liao, C-J., Chen, C-M., Lin, C-H. (2007). *Minimizing makespan for two parallel machines with job limit on each availability interval*. Journal of the Operational Research Society 58 938–47.

[23] Dell' Amico, M., Martello, S. (2001). *Bounds for the cardinality constrained P||Cmax problem*. Journal of Scheduling 4:123–38.

[24] Yang, D-L., Hsu, C-J., Kuo, W-H. (2008). *A two-machine flow shop scheduling problem with a separated maintenance constraint*. Computers & Operations Research 35:876–83.

[25] Hsu, C-J., Low, C., Su, C-T., (2010). *A single-machine scheduling problem with maintenance activities to minimize makespan*. Applied Mathematics and Computation 215: 3929–3935.

[26] Pinedo, M. (1995). *Scheduling: theory, algorithms and systems*. New Jersey: Prentice-Hall.

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