Shiraz Journal of System Management
Vol. 1, No. 3, (2013), 1-13

# A Genetic Algorithm with Modified Crossover Operator for a Two-Agent Scheduling Problem

**Maziyar Yazdani***
Department of Industrial Engineering,
University of Tehran,
Tehran, Iran

**Fariborz Jolai**
Department of Industrial Engineering,
University of Tehran,
Tehran, Iran
f.jolai@ut.ac.ir

**Abstract.** The problem of scheduling with multi agent has been studied for more than one decade and significant advances have been made over the years. However, most work has paid more attention to the condition that machines are available during planning horizon. Motivated by the observations, this paper studies a two-agent scheduling model with multiple availability constraint. Each agent aims at minimizing a function which depends only on the completion times of its jobs. The problem is to find a schedule that minimizes the objective function of one agent, subject to the objective function of the other agent does not exceed a given threshold Q. some new dominance properties for this problem percent and next, using these properties, we develop a genetic algorithm with modified crossover for the problem. Computational results are also presented to determine the performance of the proposed genetic algorithms.

**Keywords:** Scheduling; two agents; single machine; availability constraint; genetic algorithm.

Received: April (2013); Final Revision: June (2013)
*Corresponding author

# 1. Introduction

In traditional scheduling problems, many problems are solved conventionally in a one-agent environment, but many practical situations where revealed this assumption is not applicable in many real life conditions. In aspect of applications of scheduling with two competing agents (some of them focusing on game theory aspects of the problems); Curiel et al. [1] and Hamers et al. [2] studied applications in industrial management, Kim et al. [3] focused on project scheduling, Cres and Moulin [4] focused on an application in a queuing setting, and Shultz et al. [5] considered telecommunication services. Also Agnetis et al. [6] present examples of scheduling involving multiple agents competing on the usage of common processing resources in different application environments and methodological fields, such as decision theory, artificial intelligence, and operations research.

Agnetis et al. [6] and Baker and Smith [7] were the pioneers that brought the concept of multi-agent into the scheduling problem. Many research has been conducted to peruse the multi agent concept in scheduling under different machine environments and various criteria. For details on these researches, the reader may refer to [8-16].

In the other hands, most literature in scheduling problems assumes that the machines are continuously available over the planning horizon. However, this assumption may not be true in many practical situations. For instance, a machine may not be available during the planning horizon due to maintenance activities, tool changes, or breakdowns. It is clear that the maintenance activity is important to improve the quality of the products or the production efficiency of the machines. A comprehensive review of these literatures has been conducted by Schmidt [17] and Ma et al.[18].

Most of the research in scheduling with two competing agents assumes that the machines are continuously available over the scheduling horizon. However, machines might not be continuously available in many realistic situations. In this paper, we study a two-agent scheduling problem on a single machine with periodic maintenance where the objective is to minimize the total completion time of jobs from the first agent given

that the maximum tardiness and of jobs from the second agent cannot exceed an upper bound. To the best of our knowledge, no work has been done with scheduling problems in which multi agents and periodic maintenance are considered simultaneously.

Rest of paper is organized as follows; in next Section, problem definition is given. Section III presents some dominance properties used in proposed algorithm. Section IV discusses the structure of proposed algorithm. The analysis of computational experiments is provided in Section V. Finally, conclusion are presented in the last section.

## 2. Problem Description

The problem under study can be described as follows. There are $n$ jobs which are processed on a single machine. Each job belongs to either one of the two agents, namely AG1 and AG2. The processing time is shown by .

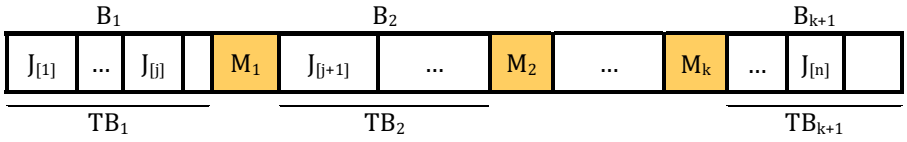


**Figure 1.** Illustration of the problem.

The machine is not continuously available for processing throughout the scheduling horizon, and it has multiple fixed and predefined unavailability periods $M_j$ is the $j^{th}$ unavailability period that $SM_j$ is the start time, and $FM_j$ is the finish time of $j^{th}$ unavailability period. If jobs between any two consecutive unavailability periods are considered a batch, a schedule can be viewed as batches of jobs separated by unavailability periods. The objective of the problem is to find a schedule that minimizes the total completion time of the jobs of AG1 with the restriction that the maximum tardiness of the jobs of agent AG2 does not exceed a given upper bound UB.

Assumptions made in this paper are as follows:

a) The machine can handle one job at a time.

b) Because of the high cost of holding the machine idle in many production environments, it is assumed that the idle insert is not allowable.

c) All jobs are ready at time zero.

d) Preemption is not allowed

e) There is no setup time required before a job is processed.

f) Time intervals between two consecutive maintenance activities are the identical and fixed.

Fig. 1 illustrates the representation of problem, where k denotes the number of unavailability periods. Bb is denoted by batch number $b(b-1, 2, ..., k+1 \ and \ k+1 \leqslant n)$.$TB_b$ is the capacity of batch b where $TB_2 - SM_b - FM_{b-1}$ where in this study is constant.

Using the three-field notation of Graham et al. [19], the investigated scheduling problem is then denoted as $1, h_k|nr-a|\sum C_i^{AG_1} : T_{Max}^{AG_2} \leqslant UB$, where 1 denotes a single machine, nr denotes nonresumable case, a denotes availability constraints,$\sum C_i^{AG_1}$ denotes makespan of jobs of AG1 and $T_{Max}^{AG_2}$ denotes sum of maximum tardiness of jobs from the second agent.

**Property 1.** Problem $1, h_k|nr-a|\sum C_i^{AG_1} : T_{Max}^{AG_2} \leqslant UB$ is strongly NP-hard.

**Proof.** Clearly, the problem is strongly NP-hard since the problem, which minimizes the makespan subject to multiple unavailability periods and non-resumable jobs($1|nr-pm|C_{Max}$) [20, 21] for one agent, is strongly NP-hard.

## 3. Dominance Properties

In this section, we present some dominance properties of the considered problem. The crossover of the proposed genetic algorithm is based on these properties. To prove these properties we need some definitions.

**Definition 1.** The slack time (ST) of a batch is defined as the amount of time unscheduled in a batch.

**Definition 2.** Let $S^r = S(i \leftrightarrow j)$ be the sequence $\acute{S}$ is obtained from sequence $S$ by a pairwise interchange of jobs $i$ and $j$ in sequence $S$ .

**Property 2.** Let $i$ and $j$ be adjacent jobs from $AG_1$ in $S$ where placed in same batch. If $p_j < p_i$ then $S$ will be dominated by $\acute{S}$ .

**Proof.** The result follows immediately from the SPT rule.

**Property 3.** Let $i$ and $j$ be adjacent jobs from $AG_1$ in $S$ where placed in two batch. If $p_j > p_i$ and $p_i + ST \geqslant p_j$, then $S$ will be dominated by $\acute{S}$ .

**Proof.** Consider to partial schedules $\pi$ and $\pi^*$ as scheduled job in $S$ and $\acute{S}$ and Let $PS$ be the partial schedule composed of the remaining jobs. Since $\pi$ has the greater makespan than $\pi^*$ , the total completion of jobs under the $S$ is no lower than that under $\acute{S}$. Thus, S is dominated by $\acute{S}$ .

**Property 3.** Let $i$ and $j$ be adjacent jobs from $AG_1$ in $S$ where placed in two batch. If $p_j > p_i$ and $p_i + ST \geqslant p_j$, then $S$ will be dominated by $\acute{S}$ .

**Proof.** The result follows immediately from the EDD order.

**Property 3.** Let $i$ and $j$ be adjacent jobs from $AG_1$ in $S$ where placed in two batch. If $p_j > p_i$ and $p_i + ST \geqslant p_j$ and $FM + p_i - d_i \leqslant T_{\max}$ ($FM$ is finishing time of maintenance activity between $i$ and $j$ ), then $S$ will be dominated by $\acute{S}$.

**Proof.** Proof is omitted since it is similar to property 3.

**Property 4.** Let $i$ and $j$ be adjacent jobs from $G_1$ in $S$ where placed in same batch. If $PS|p_j|p_i \quad d_1 \leqslant T_{\max}$, then $S$ will be dominated by $\acute{S}$.

**Proof.** Proof is straightforward and is omitted.

**Property 5.** Let $i$ and $j$ be adjacent jobs from $AG_2$ and $AG_1$ in $S$ where placed in two batch. If $\acute{S}$ be a feasible solution and if $p_j \geqslant p_i$ and $p_i + ST \geqslant p_j$ , then $S$ will be dominated by $\acute{S}$.

**Proof.** Proof is omitted since it is similar to property 3.

## 4. Proposed Genetic Algorithm

In this paper, we utilize the genetic algorithm (GA) which has been used successfully to find near optimal solutions to many complex problems .The GA usually starts with a population of feasible solutions and iteratively replaces the current population by a new population until stopping condition satisfied.

*A. Encoding*

In this study, we use a vector of integer numbers to represent a given sequence. The length of the vector is equal to the number of job. Sample of solution representation shown in Fig. 2.


**Figure 2.** Sample of solution representation.

*B. Population size*

In this research each generation has a population of 50, 100,150 chromosomes for small, medium and large size problem respectively. The initial population is generated randomly.

*C. Crossover*

Crossover is an operator to generate new offspring from two parents. In the rest of this subsection, proposed methods according to dominate properties, for crossover operator are presented. Proposed crossover is given as follows (see an example in Fig. 3):

**Step 1:** pick up two sequences, so that the first sequences is the DAD and second one is the MOM.

**Step 2:** Create empty template (ET) and assign a randomly generated number, 0 and 1 to each cell.

**Step 3:** Copy the jobs from the DAD to the locations of the rand = 1 ring to the same positions in the SON.

**Step 4:** The jobs that have already been selected from the DAD are deleted from the MOM, so that the repetition of a job in the SON is avoided.

**Step 5:** Complete the remaining empty block locations with the undeleted blocks that remain in the MOM as:

```
UB = undeleted blocks
For i = 1 to (n_A + n_B)
  if location job[i] = empty && location job[i + 1] = full
        location job[i] = UB[1]
        UB[1] = [ ]
  elseif location job[i] = empty && location job[i + 1] = empty
        if condition property 1 − 5 hold for UB[1] & UB[2]

        UB[1] & UB[2] scheduled according to propert 1 − 5
        else
            location job[i] = UB[1]
            location job[i + 1] = UB[2]
        End if
        UB[1] = [ ]
        UB[2] = [ ]
  End if
end
```

Where $n_A$ and $n_B$ is number of jobs that belong to AG1 and AG2 respectively. This procedure repeated for daughter.

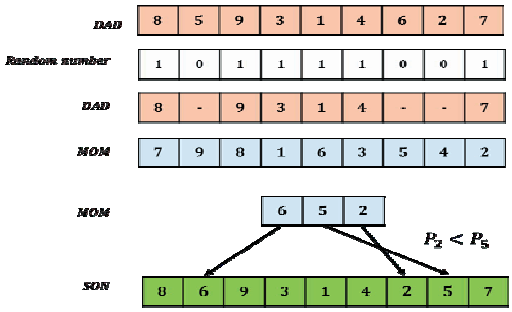


Figure 3.A proposed crossover operator. (2 and 5 are in same batch and belong to AG1 so according to property 2 since $p2 < p5$; 2 scheduled before 5)

*D. Mutation*

Mutation is another main operator to prevent fall into local optimum. For the mutation operator employed in this problem, randomly chosen two jobs and are swapped. Mutation is shown in Fig. 4.

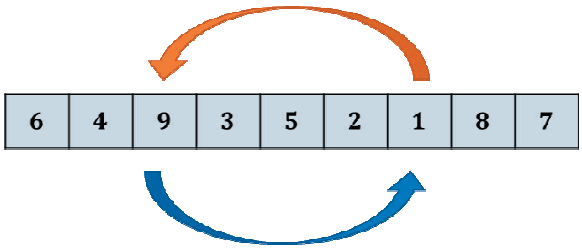


**Figure 4.** Mutation operator

*E. Selection*

It is a procedure to select offspring from parents to the next generation. In our study, Chromosomes with a lower objective function are more desirable, so the top 10% of the chromosomes are automatically copied to the next generation and rest chosen randomly.

*F. Termination*

best OFV of each iteration is recorded and compared to that obtained so far.in this study The algorithm stops, if the OFV has not been improved for the last max *iter* iterations, where max *iter* is a control parameter, in our study max *iter* equal to number of jobs

## 5. Computational Resutts and Discussion

For generating our instances, proper values should be created. Moreover, each of these parameters has special distribution function that will be described, in the following.

To present the efficiency of the proposed solution method, problems with different sizes are considered. The small size problems are associated with 8, 9 and 10 jobs, medium size with 10, 20 and 30 jobs, and large size with 60, 80 and 100 jobs. The processing times are generated from the discrete uniform distribution [2, 15] and also [5, 30] and upper bound for $T_{Max}$ for second agent is $\sum_{iEAG2p_i} + \lceil \frac{\sum_{iEAG2p_i}}{DN} \rceil \times DN$,

Capacity of every batch was determined by $TB = \alpha \times \sum p_i$ with $\alpha = 0,4,0,7$. The due dates of each job is drawn from the uniform distribution $[\sum p_i(1 - T - \frac{R}{2}), \sum p_i(1 - T + \frac{R}{2}]$. The two parameters R and T are the due date range and tardiness factor, respectively. (R=0.5 and 0.6, T=0.2 and 0.5)

*A. Experimental results*

Algorithms are coded in Matlab language and run on a personal computer with 2.2 GHz Intel Core 2 Due CPU and 2 GB of RAM memory under a Microsoft Windows 7 environment. 144 test problems with the number of jobs varying from 8 to 100 are generated. There are sixteen instances for each problem size. Each test was repeated for 5 runs due to each instance and afterwards, for evaluating the performance of these two algorithms, time of best objective functions and average normalized objective function of 5 runs are reported for each algorithm. Normalized OF is obtained from the formula:

$$normalizedOF = \frac{Alg(OF) - min(OF)}{max(OF) - min(OF) + \varepsilon}$$

As can be seen, our proposed GA with modified crossover provides better results than GA with uniform crossover. These examinations are better demonstrated in Figs. 5 ,6 and 7 in which the fluctuations of the average normalized OF and the average CPU time and average Iteration for each algorithm to stop condition satisfied are displayed in each problem size.

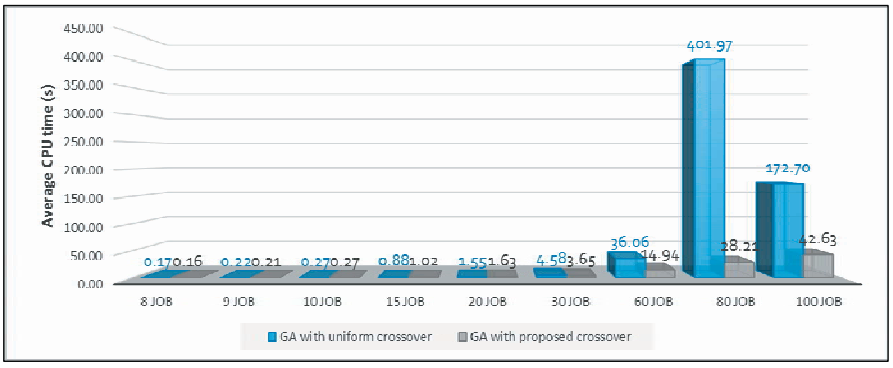


**Figure 5.** Comparative result for CPU Time which obtained by algorithms over the different problem size.(Smaller is better)

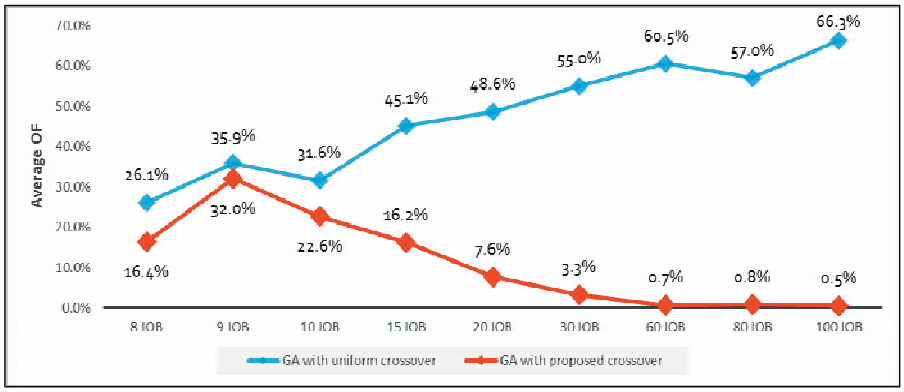


**Figure 6.** Comparative result for average normalized OF which obtained by algorithms over the different problem size. (Smaller is better)

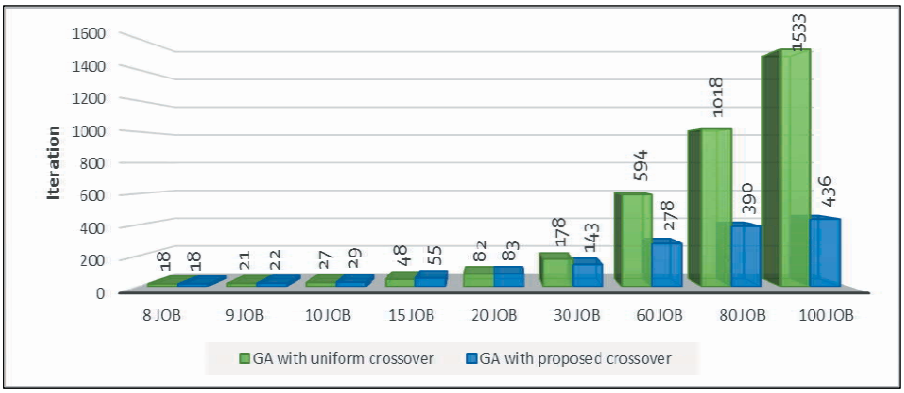


**Figure 7.** Comparative result for average Iteration for each algorithm to stop condition satisfied over the different problem size. (Smaller is better)

## 6. Conclusion

This paper addressed a two-agent single-machine scheduling problem with periodic unavailability. In this paper, we have study an problem to consider unavailability period in two agent scheduling problem scheduling to minimizing the total completion time of jobs from the first agent given that the maximum tardiness and of jobs from the second agent cannot exceed an upper bound. To tackle this NP-hard problem, we had proved several properties for our problem and then according this properties we proposed a modified crossover. To evaluate the effectiveness and robustness of the proposed algorithm we compared it against a genetic algorithm with uniform crossover and comparative results revealed the absolute superiority of our proposed algorithm.

## References

[1] Curiel, I. Pederzoli, G., and Tijs, S. (1989), "Sequencing games", European Journal of Operational Research, vol. 40, 344-351.

[2] Hamers, H., Borm, P., and Tijs, S. (1995), "On games corresponding to sequencing situations with ready times", Mathematical Programming, vol. 69, 471-483.

[3] Kim, K., Paulson Jr, B. C., Petrie Jr, C. J., and Lesser, V. R. (2000), "Compensatory negotiation for agent-based project schedule coordination", in MultiAgent Systems, 2000. Proceedings. Fourth International Conference on, 405-406.

[4] Crs, H. and Moulin, H. (2001), "Scheduling with opting out: improving upon random priority", Operations Research, vol. 49, 565-577,

[5] Schultz, D., Oh, S. H., Grecas, C. F., Albani, M., Sanchez, J., and Arbib, C. et al. (2002), "A QoS concept for packet oriented S-UMTS services", in IST Mobile and Wireless Telecommunications Summit.

[6] Agnetis, A., Mirchandani, P. B., Pacciarelli, D., and Pacifici, A. (2004), "Scheduling problems with two competing agents", Operations Research, vol. 52, 229-242.

[7] Baker, K. R. and Smith, J. C. (2003), "A multiple-criterion model for machine scheduling", Journal of Scheduling, vol. 6, 7-16.

[8] Yu, X., Zhang, Y., Xu, D., and Yin, Y. (2013), "Single machine scheduling problem with two synergetic agents and piece-rate maintenance", Applied Mathematical Modelling, vol. 37, 1390-1399.

[9] Yin, Y., Cheng, S. R., Cheng, T., Wu, W. H., and Wu, C. C. (2013), "Two-agent single-machine scheduling with release times and deadlines", International Journal of Shipping and Transport Logistics, vol. 5, 75-94.

[10] Wu, W. H. (2013), "Solving a two-agent single-machine learning scheduling problem", International Journal of Computer Integrated Manufacturing, 1-16.

[11] Wu, W. H. (2013), "A Two-Agent Single-Machine Scheduling Problem with Learning and Deteriorating Considerations", Mathematical Problems in Engineering, vol. 2013.

[12] Liu, P. and Tian, X. (2013), "Two-Agent Single-Machine Scheduling with Resource-Dependent Starting Times", Mathematical Problems in Engineering, vol. 2013.

[13] Lee, W. C., Chung, Y. H., and Huang, Z. R. (2013), "A single-machine bi-criterion scheduling problem with two agents", Applied Mathematics and Computation, vol. 219, 10831-10841.

[14] Gu, Y., Fan, J., Tang, G., and Zhong, J. (2013), "Maximum latency scheduling problem on two-person cooperative games", Journal of Combinatorial Optimization, 1-11.

[15] Fan, B., Cheng, T. E., Li, S., and Feng, Q. (2013), "Bounded parallel-batching scheduling with two competing agents", Journal of Scheduling, 1-11.

[16] Yin, Y., Wu, C. C., Wu, W. H., Hsu, C. J., and Wu, W. H. (2012), "A branch-and-bound procedure for a single-machine earliness scheduling problem with two agents", Applied Soft Computing.

[17] Schmidt, G. (2000), "Scheduling with limited machine availability", European Journal of Operational Research, vol. 121, 1-15.

[18] Ma, Y., Chu, C., and Zuo, C. (2010), "A survey of scheduling with deterministic machine availability constraints", Computers & Industrial Engineering, vol. 58, 199-211.

[19] Graham, R. L., Lawler, E. L., Lenstra, J. K., and Rinnooy Kan, A. (1977), "Optimization and approximation in deterministic sequencing and scheduling: a survey", Annals of Discrete Mathematics. vol. 5, 287-326.

[20] Low, C., Hsu, C. J., and Su, C. T. (2010), "A modified particle swarm optimization algorithm for a single-machine scheduling problem with periodic maintenance", Expert Systems with Applications, vol. 37, 6429-6434.

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