Non-Preemptive Open Shop Scheduling Considering Machine Availability

A. Shojaei Barjouei[1], Abbas Barabadi[1], R. Tavakkoli-Moghaddam[2]
[1] Department of Technology and Safety, UiT: The Arctic University of Norway, Tromsø, Norway
[2] School of Industrial Engineering, College of Engineering, University of Tehran, Tehran, Iran
(E-mail: abbas.b.abadi@uit.no)

Abstract - **This paper considers a non-preemptive open shop scheduling problem (OSSP), in which machines are not available to process jobs on known periodic interval times resulted from periodic service repair, rest period, and so on. Asymmetric transportation time between machines is considered, which can be different from one job to another. The objective is to minimize the weighted mean completion time (WMCT). Since the problem is categorized into NP-hard class, two meta-heuristic algorithms including genetic algorithm (GA) and differential evolution (DE) are proposed. Meanwhile, a new initial population is introduced, which significantly improves the performance of the algorithms. Finally, the performance of the algorithms is validated through some large-sized instances and the results are discussed.**

Keywords - **Open shop scheduling, Machine availability, Transportation time, Weighted mean completion time, Differential evolution, Genetic algorithm.**

I. INTRODUCTION

In an open shop scheduling problem, each of *n* jobs is supposed to be processed by *m* machines in arbitrary order [1]. However, machines may be unavailable due to preventive maintenance, rest period, uncompleted jobs from the previous working shift which should be processed at the beginning of the current shift, and so on [1]. OSSP provides a wide range of applications including timetable problem, manufacturing plants, optical network, communication scheduling and so on [1]. Sheikhalishahi et al. [2] illustrated a real application of OSSP considering preventive maintenance in an automobile spare parts manufactory where 9 jobs are supposed to be processed on 9 machines (i.e., 81 operations) in a pressing and forming shop.

Strusevich [3] considered a known time lag between the completion of a task and the beginning of the next task of the same job in an OSSP. Due to the actual transportation of a job between machines, he named this time lag as transportation time. Also, he referred to another interpretation of this time lag in chemical and metallurgic applications, as the cooling or heating time.

Ma et al. [4] summarized scheduling problems with availability constraints caused by preventive maintenance taking into account their complexity. Accordingly, the OSSP with availability constraint even in small cases is categorized into NP-hard class. Hence, applying approximation approaches can be more effective than exact methods. Huang et al. [5] proposed four algorithms including GA, Particle Swarm Optimization (PSO), cuckoo search algorithm, and Ant Colony Optimization (ACO) for OSSP. DE is another well-known algorithm that has rarely been implemented for OSSP; however, there are some studies using this algorithm to other scheduling problems, such as parallel machines [6].

In this paper, a non-preemptive OSSP with machine availability constraint is purposed in which machines are not available to process jobs on known periodic intervals. Available/unavailable intervals are assumed to be constant and predefined for each machine while they vary from one machine to another. Asymmetric transportation times between machines is another feature of the purposed OSSP caused by considering different routes to move between machines which reduces route interception in the shop. Moreover, different jobs have different transportation times on the same route, which can be resulted from using various vehicles to carry various parts. Furthermore, the time which a job is on a machine is divided into three parts including setup, process, and removal time. Meanwhile, WMCT is considered as the objective function, which should be minimized [7]. To solve the purposed OSSP, a new initial population is introduced which significantly improves the results of both GA and DE meta-heuristics.

The rest of this paper is organized as follows. In Section II, considering some assumptions the problem is defined. Then, encoding scheme and proposed initial populations are introduced in Section III. Sections IV and V contain GA and DE algorithms, respectively. Computational evaluation is presented in Section VI. Finally, the paper is concluded in Section VII.

II. PROBLEM DEFENITION

As shown in Fig. 1, the available times for each machine are considered like batches which jobs should be located into them while the total time of jobs does not exceed the batch time (T_j). In Fig. 1, $J_{[i]}$ indicates the job in the *i*-th position of sequence and B_{jl} is the *l*-th batch of machine j. Moreover, the available (T_j) and unavailable (t_j) intervals are constant and predefined for each machine.

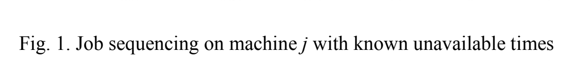

Fig. 1. Job sequencing on machine *j* with known unavailable times

A. Assumptions

The following assumptions are considered in the purposed OSSP.

1) Each machine can process at most one job at a time.
2) Each job can be processed at most on one machine at a time.
3) Jobs can be processed at any arbitrary sequence.
4) All jobs are ready to be processed at the time zero.
5) No job interruption is allowed.
6) Only one machine of each kind is in the shop.
7) Setup/removal times are dependent on jobs and machines.
8) Available/unavailable times are dependent on machines and they are constant during the planning horizon.

B. Notations

The following notations including indices, parameters and decision variables are used in this problem:

Indices

i	job indices (*i* = 1,2,…,*n*).
j, h	machine indices (*j, h* = 1,2,…,*m*).
l	batch indices (*l* = 1,2,…,*b*).

Parameters

S_{ij}	setup time of job *i* on machine *j*.
p_{ij}	process time of job *i* on machine *j*.
R_{ij}	removal time of job *i* on machine *j*.
Tr_{ijh}	travel time of job *i* from machine *j* to machine *h*.
T_j	consecutive available time length of machine *j*.
t_j	consecutive unavailable time length of machine *j*.
v_i	importance of job *i*.
w_i	weight of job *i*; $w_i = v_i / \sum_{i=1}^{n} v_i$.

Decision variables

C_{ij}	completion time of job *i* on machine *j*.
C_i	completion time of job *i*; $C_i = \sum_{j=1}^{m} C_{ij}$.

Accordingly, the objective function will be $WMCT = \sum_{i=1}^{n} w_i \, C_i$.

III. ENCODING SCHEME AND INITIAL POPULATION

A. Encoding scheme

A permutation of operations (genes) is considered as a chromosome which represents a solution. For instance, consider a shop, in which two types of shafts are supposed to be produced by two machines including lathe machine and milling machine while the sequence of operations can be arbitrary. Taking into account different sizes and substances of the raw materials, setup, process, and removal times vary from one product to another. Moreover, the positions of the machines and materials flow direction in the shop lead to asymmetric transportation times. For such an OSSP with 2 jobs and 2 machines, the chromosome below is a possible solution, where O_{ij} is the operation related to job *i* on machine *j*.

O_{11}	O_{12}	O_{21}	O_{22}
1	3	4	2

Fig. 2 illustrates a schematic view of the above example, in which while job 1 (product 1) is being processed on machine 1 (O_{11}), job 2 (product 2) is on machine 2 (O_{22}). Thereafter, job 1 goes to machine 2 (O_{12}) to be finalized and job 2 goes to machine 1 (O_{21}) as its last operation.

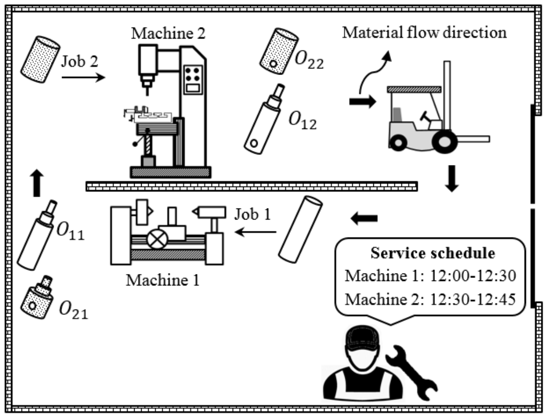

Fig. 2. Schematic view of the OSSP considering availability due to periodic maintenance and asymmetric transportations

B. Initial population

Two kinds of initial populations consist of the pure random population (PRP) and semi-guided population (SGP) are described following. In the PRP, a random permutation of the set $\{1,2,\ldots,n{\times}m\}$ is considered as an individual, where $n{\times}m$ is the number of operations. In the SGP; however, some guides to improve WMCT are considered. For instance, locating more important jobs earlier in the sequence will considerably improve WMCT. The proposed SGP will be obtained through the following steps:

Step 1. While the stopping criterion is not met, repeat steps 2 to 8.
Step 2. Set $Turn = 1$.
Step 3. Assign the *m* first highest important jobs to the *m* machines randomly as each machine processes exactly one of the jobs.
Step 4. While all of the *n* jobs are not assigned $Turn$ time(s) to any machine, repeat steps 5 to 7.
Step 5. Update the release time of machines.
Step 6. Select the highest important job which is not assigned more than $Turn - 1$ time(s).
Step 7. Considering travel times and unavailable times, assign the selected job in step 6 to a machine as the job starts at the earliest possible time.

Step 8. If $Turn > m$ then go to step 9. Otherwise, put $Turn = Turn + 1$ and go to step 4.
Step 9. Save the completed chromosome in an archive.
Step 10. Check the stopping criterion which is producing several chromosomes.

IV. GENETIC ALGORITHM

GA is a very powerful and popular guided random search techniques for solving optimization problems. GA is an evolutionary algorithm (EA) which simulates the natural evolution of asexual species to search solution space. Accordingly, a new offspring is created by combining two parents and the crossover operator is applied as a fundamental component of this search technique. GA parameters are the probability of crossover (p_c), probability mutation (p_m), number of individuals in each generation (*popsize*), and stopping criterion. A review of this algorithm is presented by Goldberg [8].

In the following, some basic terms of GA are discussed and then the structure of our proposed GA is defined.

A. Fitness function

Although the objective is minimization, to simplicity, we transform it to maximization form as the fitness function by subtracting the objective value from a large positive number *M*. Hence, the fitness of individual *k* (f_k) is calculated as $f_k = M - WMCT_k$, where $WMCT_k$ is the objective function value of individual *k*. To calculate the $WMCT$, the algorithm starts from the first operation in the sequence, assigns the operation to the related machine, determines the earliest possible time at which the operation can be processed on the machine based on the travel time from the previous machine and unavailable intervals as well as the release time of the machine from processing operations in the earlier sequences, updates the release time of the machine, which is, in fact, the completion time of the current operation and follows this procedure for the next operation in the sequence.

B. Selection

To select parents to be muted together the roulette wheel scheme is used; in which individuals with higher fitness value have more chance to be selected. Based on the roulette wheel strategy, two parameters including the probability of selecting (PS_i) and cumulative probability (CP_i), should be calculated for individual *i*, which are achieved through (1) and (2), respectively.

$$PS_i = \frac{f_i}{\sum_{j=1}^{popsize} f_j} \tag{1}$$

$$CP_i = \sum_{j=1}^{i} PS_j \tag{2}$$

Then, a random number *r* is chosen from [0 1] and the *i*-th individual satisfying the condition $CP_{i-1} < r \le CP_i$, is selected as one of the two parents. This process is repeated to find the second parent.

C. Crossover

To combine two selected parents and generate two offspring, 2-point crossover operator is applied with the probability of p_c. The 2-point crossover operator is defined through the following four steps (See [9] for more details):

Step 1. Select two genes of parent 1 at random.
Step 2. Copy all the genes between two selected genes from parent 1 to offspring 1 exactly in the same positions.
Step 3. Find the genes which do not exist in offspring 1 from parent 2 in order from left to right and fill the empty positions of offspring 1 from left to right.
Step 4. Repeat Steps 1 to 3 to produce offspring 2, but reverse the role of parents 1 and 2.

D. Mutation

To avoid falling in local optima, the mutation operator is applied to each offspring with the probability of p_m. Accordingly, two genes are selected at random and their positions will be exchanged (see [9] for more details).

E. Stopping criterion

The algorithm continues and new generations are produced until the stopping criterion, which is $n \times m \times \theta$ seconds from the start of the algorithm is met. Where, *n*, *m* and θ are number of jobs, number of machines and time coefficient, respectively. Consequently, more time for larger problems will be considered.

F. Proposed GA

The proposed GA works through the following steps:
Step 1. Generate *popsize* individuals as initial population using PRP or SGP.
Step 2. Compute the fitness of each individual.
Step 3. Apply roulette wheel to select a pair of individuals as parents.
Step 4. Apply the crossover operator with the probability of p_c and generate two offspring.
Step 5. Apply mutation operator with the probability of p_m for each offspring.
Step 6. Compute the fitness of each offspring.
Step 7. Repeat steps 3 to 6 till the *popsize* offspring is produced.
Step 8. Sort all new and old individuals based on their fitness value in descending order (2×*popsize* individuals).
Step 9. Copy the *popsize* number of the best individuals to the next generation.
Step 10. Repeat Steps 3 to 9 while the *stopping criterion* is not met.

V. DIFFERENTIAL EVOLUTION

DE is one of the modern EAs which was defined by Storn and Price during their attempts to solve the Chebychev polynomial fitting problem [10]. DE utilizes mutation and crossover operators and a selection strategy while they are different from those of the GA. According to DE, a trial vector of a solution is generated by adding a weighted difference vector between two vectors to a third vector. Thereafter, the crossover operator is applied. Then, if the resulted vector improves the objective function, the old vector will be replaced by the newly generated vector.

A. Mutation

The mutation operator in DE generates a new child by adding the scaled difference of two parents to a third one as shown in (3).

$$U(\delta,i,j) = chrom(\gamma,i,j) + F * \left(chrom(\alpha,i,j) - chrom(\beta,i,j)\right) \quad (3)$$

where $\alpha \neq \beta \neq \gamma \neq \delta$ are selected randomly from $\{1,2,\dots,popsize\}$. α,β,γ are the indices of individuals related to the current generation and δ is the index of a new individual for the next generation. So, $chrom(\alpha,i,j)$ indicates O_{ij} in α-th individual of the current generation. F is the scale parameter which controls enlarging of differential variation. However, the mutation operator usually generates an illegal individual. Therefore, a repair procedure based on integer order criterion (IOR) is used [11]. According to IOR, the smallest gene in the illegal chromosome is set to 1, the second gene evaluated as 2 and consequently, the last one will be $n \times m$ (See [11] for more details).

B. Crossover

After mutation, the crossover operation is applied with the probability of CR according to (4).

$$trial(\delta,i,j) = \begin{cases} U(\delta,i,j) & \text{if } rand_{ij} < CR \text{ or } (i,j) = (i_r, j_r). \\ chrom(\delta,i,j) & \text{otherwise.} \end{cases} \quad (4)$$

where $rand_{ij}$ is randomly chosen from interval [0 1], i_r and j_r are randomly selected from the indices of jobs and machines in a row. Meanwhile, $chrom(\delta,i,j)$ indicates O_{ij} in the δ-th individual of the current generation.

C. Selection

After applying the mutation and crossover operators, to guarantee the improvement of individuals in each generation, the generated trials will be transferred to a new generation if and only if they improve the current generation. The selection procedure is defined in (5) where $f(Z)$ indicates the fitness of *Z* and *G* refers to the current generation.

$$chrom(\delta,i,j)_{G+1} = \begin{cases} trial(\delta,i,j) & \text{if } f(chrom(\delta,i,j)) < f(trial(\delta,i,j)). \\ chrom(\delta,i,j)_G & \text{otherwise.} \end{cases} \quad (5)$$

D. Stopping criterion

The same stopping criterion described for the GA used for DE. Therefore, the DE algorithm continues $n \times m \times \varepsilon$ seconds from the start of the generating procedure.

E. Proposed DE

The structure of the proposed DE is briefly outlined in Fig. 3.

```
Proposed DE

    $G = 0$
    Generate $popsize$ individuals as initial population using PRP or SGP.
    Evaluate the fitness of each individual.
    Set the parameters $CR$ and $\varepsilon$.
    While $stopping\ criterion\ is\ not\ met$ do
      Set the parameter $F$.
      For $\delta = 1$ To $popsize$
        Select $\alpha \neq \beta \neq \gamma \neq \delta$ randomly.
        Select $i_{rand} \in \{1,2,\dots,n\}$ and $j_{rand} \in \{1,2,\dots,m\}$ randomly.
        For $i = 1$ To $n$
          For $j = 1$ To $m$
            $U(\delta,i,j) = chrom(\gamma,i,j)_G + F * (chrom(\alpha,i,j)_G - chrom(\beta,i,j)_G)$
            If $rand_{ij} \in [0\ 1] < CR$ or $(i,j) = (i_{rand}, j_{rand})$ Then
            $trial(\delta,i,j) = U(\delta,i,j)$
            Else
            $trial(\delta,i,j) = chrom(\delta,i,j)_G$
            End If
          Next
        Next
        Call IOR
        If $f(chrom(\delta,i,j)) < f(trial(\delta,i,j))$ Then
        $chrom(\delta,i,j)_{G+1} = trial(\delta,i,j)$
        Else
        $chrom(\delta,i,j)_{G+1} = chrom(\delta,i,j)_G$
        End If
      Next
      $G = G + 1$
    End While
```

Fig. 3. Outline of the proposed DE

VI. COMPUTATIONAL EVALUATION

The relative percentage deviation (RPD) is used as a popular performance measure taken from (6) to investigate the performance of different combinations of the proposed algorithms and initial populations (i.e., DE_{PRP}, DE_{SGP}, GA_{PRP} and GA_{SGP}) [12]. To this aim, four sets of large-sized instances are generated at random in which $n = 30{,}40$ and $m = 5{,}10$. Considering five replications for each combination of (n,m), we have 20 instances. The number of batches is regarded as $b = 2n$. Furthermore, the parameters of the problem are the processing times with a uniform distribution in the interval $(1, 99)$, the setup and removal times taken to be uniform in range $(1, 25)$, the importance of jobs with uniform distribution over $(1, n)$, the travel times considered to be uniformly distributed in range $(1, 20)$, the consecutive unavailable time lengths with a uniform distribution in range $(1, 50)$ and consecutive available time lengths $T_j = \max\left\{a \sum_{i=1}^{n}(S_{ij} + p_{ij} + R_{ij}), \max_i(S_{ij} + p_{ij} + R_{ij})\right\}$, where a is selected randomly from $\left\{{}^1\!/_5, {}^1\!/_4, {}^1\!/_3\right\}$ [12-13]. Accordingly, the objective function will be $WMCT = \sum_{i=1}^{n} w_i C_i$; where $w_i = v_i / \sum_{i=1}^{n} v_i$.

$$RPD = \frac{Alg_{sol} - Min_{sol}}{Min_{sol}} \times 100 \quad (6)$$

where Alg_{sol} is WMCT of an instance achieved from a given algorithm and Min_{sol} is the lowest WMCT for the instance obtained by any of the algorithms.

To set the parameters of the algorithms different levels of each are examined and all the instances are solved five times with different combinations. The GA parameters are assumed as probabilities of crossover and mutation: $p_c, p_m \in \{0.85, 0.9, 0.95^*\}$, population size: $popsize \in \{20, 50, 80^*\}$, and the coefficient of time: $\theta \in \{0.2, 0.3^*, 0.4\}$, where the starred ones provide the best combination. In the same way, the DE parameters are assumed as a parameter of scale: $F \in [0.3\ 0.9]$, probability of crossover: $CR \in [0.8\ 1]$, population size: $popsize \in \{20, 50, 80^*\}$, and the coefficient of time: $\varepsilon \in \{0.2, 0.3, 0.4^*\}$. Therefore, CR is different for each problem and F changes in each generation.

The algorithms are programmed by Visual Basic.Net 2008 and run on a 2.4 GHz Intel® Core™ i5 CPU and 4GB of RAM. All of the 20 large-sized instances are solved five times by all combinations of algorithms and initial populations (i.e., DE_{PRP}, DE_{SGP}, GA_{PRP} and GA_{SGP}). To compare these algorithms, the results transformed into the RPD and the average for each problem size is indicated in Table I. Accordingly, the significant improvement by using SGP is evident, while GA and DE show similar performances. Meanwhile, the average computational times of the algorithms in seconds for each problem size is shown in Table II, where the GA works moderately faster except in the largest case. The computational time is not significantly influenced by the initial population. Also, the sensitivity of all combinations of algorithms and initial populations toward job importance (v_i) were investigated. However, no significant sensitivity was observed either in the computational time or in the RPD. The only case that influences on RPD of DE_{PRP} and GA_{PRP} is the case in which all v_is are supposed to be equal (i.e., no weight in the objective function). Accordingly, the average RPD of both DE_{PRP} and GA_{PRP} will be 0.42 which is around 50 percent better than weighted scenario. However, since SGP is based on job importance and weight, considering equal weights while applying SGP seems unreasonable.

TABLE I
AVERAGE RPD FOR THE LARGE-SIZED INSTANCES

$n.m.b$	DE_{PRP}	DE_{SGP}	GA_{PRP}	GA_{SGP}
30.5.60	0.71	0.01	0.70	0.01
30.10.60	0.96	0.02	0.98	0.02
40.5.80	0.80	0.01	0.80	0.01
40.10.80	0.98	0.02	1.02	0.02
Average	0.86	0.015	0.87	0.015

TABLE II
AVERAGE COMPUTATIONAL TIME IN SECONDS

$n.m.b$	DE_{PRP}	DE_{SGP}	GA_{PRP}	GA_{SGP}
30.5.60	64.8	66.2	50.8	50.6
30.10.60	145.6	145.6	128.2	130.6
40.5.80	82.6	82.4	75	74.4
40.10.80	174	172.6	216	215

VII. CONCLUSION

In this paper, a new initial population namely SGP was introduced, which considerably improves the performance of the two powerful meta-heuristics including GA and DE to deal with the non-preemptive OSSP with the machine availability constraint and asymmetric transportation times. To investigate the performance of the algorithms with two initial populations including SGP and PRP, 20 large-sized instances were generated at random and the results transformed into the RPD. Accordingly, although both GA and DE have similar performances, SGP makes both algorithms much more effective and robust. Moreover, the computational time is not significantly influenced by the initial population.

REFERENCES

[1] J. Y. T. Leung, *Handbook of scheduling: Algorithms, models, and performance analysis*, Chapman & Hall/CRC Computer and Information Science Series, 2004.

[2] M. Sheikhalishahi, N. Eskandari, A. Mashayekhi, A. Azadeh, "Multi-objective open shop scheduling by considering human error and preventive maintenance," *Applied Mathematical Modelling*, vol. 67, pp. 573-587, 2019.

[3] V. A. Strusevich, "A heuristic for the two-machine open-shop scheduling problem with transportation times," *Discrete Applied Mathematics*, vol. 93, no. 2-3, pp. 287-304, 1999.

[4] Y. Ma, C. Chu, C. Zuo, "A survey of scheduling with deterministic machine availability constraints," *Computers & Industrial Engineering*, vol. 58, no. 2, pp. 199-211, 2010.

[5] Z. Huang, Z. Zhuang, Q. Cao, Z. Lu, L. Guo, W. Qin, "A survey of intelligent algorithms for open shop scheduling problem," *The 11th CIRP Conference on Industrial Product-Service Systems*, vol. 83, pp. 569-574, 2019.

[6] G. Deng, K. Zhang, X. Gu, "A hybrid discrete differential evolution algorithm to minimise total tardiness on identical parallel machines," *International Journal of Computer Integrated Manufacturing*, vol. 26, no. 6, pp. 504-512, 2013.

[7] H. Kellerer, M. A. Kubzin, V. A. Strusevich, "Two simple constant ratio approximation algorithms for minimizing the total weighted completion time on a single machine with a fixed non-availability interval," *European Journal of Operational Research*, vol. 199, no. 1, pp. 111-116, 2009.

[8] D. E. Goldberg, *Genetic algorithms in search, optimization and machine learning*, Addison–Wesley: Wokingham, UK, 1989.

[9] M. E. Matta, "A genetic algorithm for the proportionate multiprocessor open shop," *Computers & Operations Research*, vol. 36, no. 9, pp. 2601-2618, 2009.

[10] R. Storn, K. Price, *Differential Evolution: A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces*, TR-95-012, International Computer Science Institute: Berkeley, California, 1995.

[11] C. Erbao, L. Mingyong, "A hybrid differential evolution algorithm to vehicle routing problem with fuzzy demands," *Journal of Computational and Applied Mathematics*, vol. 231, no. 1, pp. 302-310, 2009.

[12] B. Naderi, M. Zandieh, "Modeling and scheduling no-wait open shop problems," *International Journal of Production Economics*, vol. 158, pp. 256-266, 2014.

[13] C. J. Hsu, C. Low, C. T. Su, "A single-machine scheduling problem with maintenance activities to minimize makespan," *Applied Mathematics and Computation*, vol. 215, no. 11, pp. 3929–3935, 2010.