Send Orders for Reprints to reprints@benthamscience.ae

60 *The Open Automation and Control Systems Journal*, 2017, *9*, (Suppl-1, M6) 60-74

BENTHAM OPEN

CrossMark

The Open Automation and Control Systems Journal

Content list available at: www.benthamopen.com/TOAUTOCJ/

DOI: 10.2174/1874444301709010060

RESEARCH ARTICLE

Investigations on Genetic Algorithm Performances in a Parallel Machines Scheduling Environment

Fayçal Belkaid[1,2,*], Farouk Yalaoui[3] and Zaki Sari[1]

[1]*MELT, University of Tlemcen, Tlemcen, Algeria*
[2]*LGIPM, University of Lorraine, Metz, France*
[3]*ICD-LOSI (UMR-STMR CNRS 6281), University of Technology of Troyes, Troyes, France*

Received: October 30, 2016 Revised: February 15, 2017 Accepted: March 30, 2017

Abstract:

Objective:

The objective is to propose a resolution method to solve the identical parallel machines scheduling problem with non-renewable resources in manufacturing environment to minimize the total completion time.

Introduction:

Since the consideration of consumable resources becomes one of the strategic competitive tools to ensure companies performance and the stability of their production systems. This study considers a parallel machines scheduling problem with non-renewable resources.

Materials and Methods:

TA mathematical model is developed in order to find an optimal solution. Due to the problem complexity and prohibitive computational time to obtain an exact solution, a genetic algorithm is proposed and several heuristics are adapted to minimize the total completion time.

Results:

The simulation results show that the proposed genetic algorithm gives the same results as the mathematical model for small instances (exact solution) and performs the best compared to heuristics for medium and large instances.

Conclusion:

The scheduling problem in parallel machine environment with consumables resources is studied in this paper. A mathematical model and a metaheuristic are proposed to solve it in order to minimize the total completion time. The simulation results demonstrate the effectiveness of the proposed metaheuristic.

Keywords: Genetic algorithm, Heuristics, MILP, Parallel machines scheduling, Consumable resources, Total completion time.

1. INTRODUCTION

The production scheduling is unquestionably one of the most important parameters that affect the performance achieved. In production system, scheduling problem consists in organizing the execution time of interdependent operations using available resources in limited quantities to achieve a production plan [1]. Production scheduling problems can be classified into several families based on several aspects, and parallel machine scheduling problem is

* Address correspondence to this author at University of Tlemcen, Tlemcen, Algeria & LGIPM, University of Lorraine, Metz, France, Tel: 00213555880085; Fax: 00213-43411191; Emails: f_belkaid@yahoo.fr; faycal.belkaid@mail.univ-tlemcen.dz

one of the most examined problem in scheduling theory and represents an important research field. This importance is further amplified by the variety of complex configuration that can be reduced to this type of problem. In this regard, a vast amount of papers have appeared in the literature about this class of manufacturing model [2]. Theoretical scheduling models are limited to sequence jobs in machines. Generally, the models studied are placed in the context where resources are always available and ignore all constraints related to the availability of resources.

Nevertheless, in real industrial environment, manufacturing companies encounter resource constraints and jobs need a set of non-renewable resources such as raw materials, during the time to be processed. Furthermore, the consumable resources may be subject to some unavailability periods for various reasons which complicate the scheduling problem. In this context, the main objective of industrials is to exploit effectively their manufacturing systems to improve their reactivity and productivity by using all available resources in an optimal way. Indeed, the management of resources remains an inherent phase and it is considered as one of the important strategic tool because it provide many benefits, such as improving production rate and enhancing the system performances.

In production environments, the scheduling problem is well addressed by heuristics and scheduling algorithms, but many of them do not consider the impact of the consumable and renewable resources and situation which appear frequently can affects significantly the productivity and system behavior. Therefore, in order to consider exploit the system effectively and to improve their performances, the scheduling decisions in production shop should consider all assumptions and constraints. It can be noticed that scheduling problems in such systems are so difficult because many constrains can be defined and several parameters can be considered under various perturbations. Consequently, scheduling problems in these production systems are generally NP Hard and there are no effective methods making it possible to solve all the cases optimally [3]. Furthermore, using mathematical analytical methods seems very difficult to solve this kind of problems

This paper which is an extension of works conducted in the literature deals with a simulation study of parallel machines scheduling problem with consumable resources. It aims to present a mathematical model in order to find the appropriate assignment of jobs to machines and to define the suitable sequencing of jobs assigned to each machine with the consideration of resources requirement. Additionally, due to the complexity situation, we propose a metaheuristic based on genetic algorithm to address this problem. Subsequently, we study the behavior of several local searches with this metaheuristic in order to improve it performances.

The remainder of this paper is organized as follows. The next section presents the literature review of the considered problem. Section 3 describes the specificity of the system and the model studied in this research work. Section 4 presents the proposed mathematical model. Then, the resolution approach developed in this study is presented in section 5. Section 6, define the heuristics procedures adapted to the considered problem. The simulation results are presented in section 7. Finally, the conclusion is offered in section 8.

2. LITERATURE REVIEW

Problems related to the scheduling with consumables resources have attracted considerable attention in the scheduling community for its high impact in productivity improvement, furthermore, such a problem is often encountered in production environment, for example, in manufacturing operations, the job treatment may depend on some raw materials requirement, (*i.e.* non-renewable resources) as well as machines (*i.e.* renewable resources). Moreover, these resources have to be available at the same time in order to process jobs. Hence, the suitable organization of resources can improve the production efficiency and provide many advantages. The impact of availability of both types of resources on the complexity of such problems is best illustrated in [4], in which authors confirm that substantially all problems on parallel machines with non-renewable resources constraint are NP-hard (Non-deterministic Polynomial-time Hard).

Recently, there has been an increasing interest in single machine problems with consumable resources. Toker *et al.* [5] treat the single machine problem with a single financial resource, they show that this problem is equivalent to a two machine flow shop without financial resources; they applied the Johnson algorithm for minimizing the makespan. Xie [6] consider a single machine scheduling problem with multiple financial resource constraints, the author reduce this problem to the two machine flow shop scheduling problem. He shows that the LPT rule gives an optimal solution to the problem if the financial resources are consumed uniformly by all the jobs. Kaspi and Shabtay [7] treat a single machine scheduling problem where the processing time is defined by a convex decreasing resource consumption function and processing times depend on a common limited resource. The objective is to determine the optimal job permutation and

the resource allocation. Lazarev and Werner [8] address the scheduling problem on a single machine; they assume that processing times and the due dates are oppositely ordered. They give some polynomials solvable special cases for minimizing total tardiness. Carrera *et al.* [9]authors discuss the problem in a single machine with consumable resources and fixed delivery dates. The authors conduct a study on the complexity of these problems and they show that they are NP-hard. They use a branch and bound to solve them.

These types of problems are relatively complex and several resolution methods may be impractical in many cases of these scheduling problems. Given the success of metaheuristics in the context of solving NP-hard problems and wherein production systems are complex, many researchers have employed them to solve parallel machine scheduling problems because of their good results proved in the resolution of several complex problems, but the majority of research are focused in the makespan objective.

Daniels *et al.* [10] treat a parallel machine scheduling problem with flexible-resource in which the processing time of each job is a function of the amount of affected resource. They explore different heuristics and define two tabu-search methods in order to develop an effective schedule to minimize the makespan. Daniels *et al.* [11] investigate a parallel-machine flexible-resource scheduling problem in which job assignment to machines are not specified (UPMFRS). They develop a heuristic which consist to determine (approximately) the job allocation to machines and the affectation of resources to jobs. Subsequently, they present a decomposition heuristic and an efficient tabu-search method.

Min and Cheng [12] proposes a genetic algorithm for minimizing the makespan on identical parallel machines scheduling problem, which is efficient for large scale problems. The obtain results are compared against simulated annealing and other available heuristics to demonstrate the good performance of genetic algorithm. Serifoglu and Ulusoy [13] study a parallel machines scheduling problem with independent jobs to minimize earliness-tardiness. They propose a genetic algorithm to solve this problem. Mendes *et al.* [14] consider a parallel machines scheduling problem. The authors propose a taboo search and memetic approach based local search procedures to solve this problem. Fowler *et al.* [15] propose a hybrid genetic algorithm for a parallel machines scheduling problem to minimize the makespan with sequence dependent setups. The first step of the hybrid genetic algorithm consists to assign jobs to machines and the second step consists in dispatching rules for scheduling the individual machines. Wilson *et al.* [16] treat a parallel machines scheduling problem with a common batch setup time. To solve this problem, the authors propose a batch splitting and a genetic algorithm. A heuristic is integrated in the genetic algorithm to improve the results.

Li *et al.* [17] examine the identical parallel machine scheduling problem with controllable processing times. They define two kinds of machines, critical and non-critical machine. The processing time of jobs is linear decreasing functions of the consumed resource and the total resource consumption is limited. The authors propose a metaheuristic based simulated annealing to minimize the makespan. Lee *et al.* [18] investigate a scheduling problem in the practices of emergency logistics that deliver medical services to the affected areas after a disaster. They demonstrate that when the amount of consumable resource available at time zero is sufficient, then the problem becomes equivalent to a parallel machine scheduling problem with release dates and sequence dependent setup times; and as objective the total weighted tardiness. Belkaid *et al.* [19] investigate a scheduling problem on identical parallel machines to minimize the makespan. Each job depends on the amount of consumed resources and is characterized by different resource requirements. They propose a genetic algorithm and compare it with an exact method for small size problems and with a heuristic for large size problems. Belkaid *et al.* [20] consider the same system cited above, they propose at first, a mathematical model and at second a genetic algorithm to solve this problem. Subsequently, they suggest a local search procedure to improve the results.

Despite the riches of research works in the field of production scheduling; the literature review analysis demonstrates that a lack of studies on metaheuristics application for solving the identical parallel machines scheduling problem with non-renewable resources in manufacturing environment and not much progress has been made to minimize the total completion time. In this direction, it appears necessary to further develop and extend the research area regarding the impact analysis of theses constrains on system performances in order to optimize the total completion time. Therefore, this paper focuses on a simulation based research study in this direction.

3. PROBLEM DESCRIPTION

This investigation considers a parallel machines scheduling problem in order to analyze the impact of the studied scheduling approaches. The system studied is composed of m parallel machines and has the following characteristics:

- Each job *j* has a processing time p_j.
- Each job consumes *k* components at the beginning of its execution and can be executed on a machine *i*, when all necessary components are available.
- The components are delivered by suppliers at different times.
- The arrival of resources is represented by a curve-shaped staircase.
- Machines can process only one job at a time and they are available from time *t=0*.
- All machines have the same speed and capacity of processing.
- The preemption is not allowed in jobs processing.
- Storage buffers are located before each machine. In this research, each job is affected to the first available machine (which contains the least amount of jobs in queues). Moreover, the first arrived job at each stage is the first treated. The configuration of the studied system is specified in Fig. (**1**).

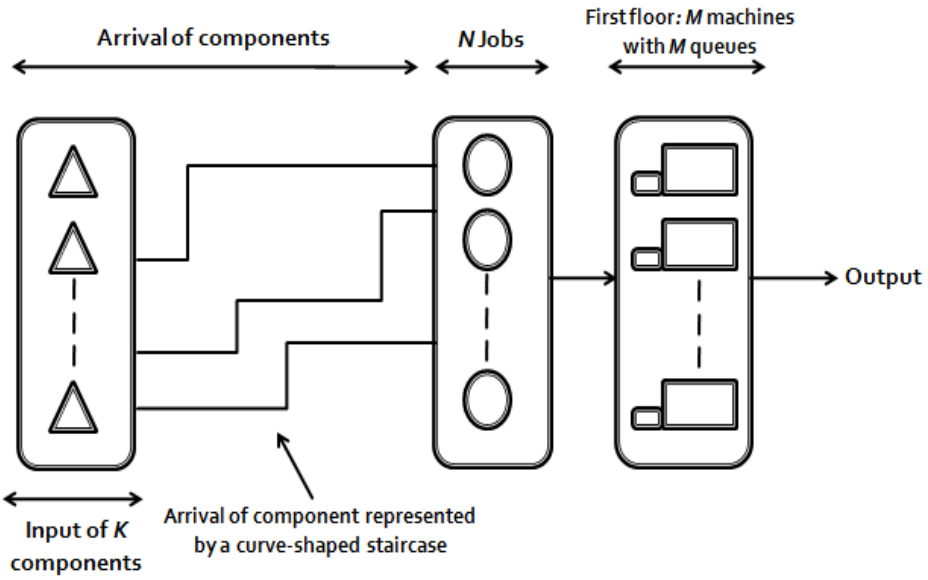

Fig. 1. Problem representation [19].

- The aim of this study is the minimization of the total completion time.

4. PROBLEM FORMULATION

In this work, a detailed Mixed Integer Linear Programming mathematical formulation for the scheduling problem is developed. Considering that the number of renewable and consumables resources are variables, the scheduling formulation represents a great challenge from the modeling point of view. In this direction, this section aims to presents a formulation based on the modeling concept of position variables. Note that there are T periods and within each period certain amount of components will arrive. To formulate the problem, the following notations are introduced:

- *n*: number of jobs.
- *m*: number of machines.
- *c*: number of components.
- T_1 : time of the first arrival.
- T_{last} : time of the last arrival.
- *j*: index of the task, j = 1, ..., n.
- *i*: index of the machine, i = 1, ..., m.
- *k*: index of the resource, i = 1, ..., c.
- n_i: number of tasks assigned to machine i.
- *p*: position of job in a machine, p=1,…n_i.
- p_j: operating time of job j.
- d_{i0}: start date scheduling.
- *g*: compute the total completion time after the execution of each job.

- d_{ip} : start date of job processing on machine i in position p.
- p_{ip}: processing time of job i on the machine in position p.
- s_{ip} : start date of job processing on machine i in position p.
- p_{ip}: processing time of job on machine i in position p.
- f_{ip}: completion date of job processing on machine i in position p.
- c_{jk}: amount of components k consumed by the job j.
- c_{ipk}: amount of component k consumed by the job in position p on machine i.
- A_{tk}: total component k arrival until time t.
- Z: a big coefficient called big Z coefficient. It's best to try to keep big Z values as small as reasonably possible in order to maintain the effectiveness of the solver.
- *TCT*: Total Completion Time.
- X_{jip} and Y_{ipt} : decision variables.

- **Objective function:**

$$Min\ TCT \tag{1}$$

- **Contraints:**

$$\sum_{j=1}^{n} X_{jip} = 1 \quad \forall\, \mathrm{i} = 1,2,\dots,\mathrm{m} \quad \forall\, \mathrm{p} = 1,2,\dots,n_i \tag{2}$$

$$\sum_{\mathrm{i}=1}^{\mathrm{m}} \sum_{\mathrm{p}=1}^{n_i} X_{jip} = 1 \quad \forall\, \mathrm{j} = 1,2,\dots,\mathrm{n} \tag{3}$$

$$p_{ip} = \sum_{j=1}^{n} X_{jip} p_j \quad \forall\, \mathrm{i} = 1,2,\dots,\mathrm{m} \quad \forall\, \mathrm{p} = 1,2,\dots,n_i \tag{4}$$

$$d_{i0} = 0 \quad \forall\, \mathrm{i} = 1,2,\dots,\mathrm{m} \tag{5}$$

$$g = 0 \tag{6}$$

$$d_{i(p-1)} + p_{ip} \leq f_{ip} \quad \forall\, \mathrm{i} = 1,2,\dots,\mathrm{m} \quad \forall\, \mathrm{p} = 1,2,\dots,n_i \tag{7}$$

$$d_{ip} = f_{ip} \quad \forall\, \mathrm{i} = 1,2,\dots,\mathrm{m} \quad \forall\, \mathrm{p} = 1,2,\dots,n_i \tag{8}$$

$$g = g + f_{ip} \tag{9}$$

$$s_{ip} = f_{ip} - p_{ip} \quad \forall\, \mathrm{i} = 1,2,\dots,\mathrm{m} \quad \forall\, \mathrm{p} = 1,2,\dots,n_i \tag{10}$$

$$c_{ipk} = \sum_{j=1}^{n} X_{jip} c_{jk} \quad \forall\, \mathrm{i} = 1,2,\dots,\mathrm{m} \quad \forall\, \mathrm{p} = 1,2,\dots,n_i \quad \forall\, \mathrm{k} = 1,2,\dots,\mathrm{c} \tag{11}$$

$$\sum_{\mathrm{i}=1}^{\mathrm{v}} \sum_{\mathrm{p}=1}^{\mathrm{w}} c_{ipk} \leq \sum_{t=T_1}^{T_{last}} A_{tk} * Y_{iwt} \quad \forall\, \mathrm{v} = 1,2,\dots,\mathrm{m} \quad \forall\, \mathrm{w} = 1,2,\dots,n_i \; \forall\, \mathrm{k} = 1,2,\dots,\mathrm{c} \tag{12}$$

$$\mathrm{Z} * (Y_{ipt} - 1) \leq s_{ip} - \mathrm{t} \quad \forall\, \mathrm{i} = 1,2,\dots,\mathrm{m} \quad \forall\, \mathrm{p} = 1,2,\dots,n_{il} \quad \forall\, \mathrm{t} = t_1,\dots,T_{last} \tag{13}$$

$$\mathrm{TCT} \geq \mathrm{g} \quad \forall\, \mathrm{i} = 1,2,\dots,\mathrm{m} \quad \forall\, \mathrm{p} = 1,2,\dots,n_i \tag{14}$$

$$X_{jip} = \begin{cases} 1 \text{ if the job j is scheduled in position p on machine i} \\ 0 \text{ otherwise} \end{cases} \tag{15}$$

$$Y_{ipt} = \begin{cases} 1 \; if \; s_{ip} \geq \mathrm{t} \\ 0 \text{ otherwise} \end{cases} \quad \textbf{(16)}$$

The function (1) is the objective function. It consists to minimize the total completion time. Constraint (2) indicates that each job j is assigned to only one machine *i* at position p. Constraint (3) ensures that each job is scheduled exactly once. Constraint (4) determines the processing time of the job in position p on machine i.

Constraint (5) makes sure that the starting time of the schedule is at time zero. Constraint (6) ensures that the total completion time is equal to zero at the starting time of the schedule. Constraint (7) aims to compute the completion time of jobs at each position p. Constraint (8) aims to update the start date of job processing on machine i in position p. Constraint (9) aims to compute the total completion time after the execution of each job in position p. Constraint (10) calculates the starting time of the job which is in position p on machine i.

Constraint (11) permits to determine the quantity of components consumed by the job scheduled in position p on machine i. Constraint (12) ensures that the amount of components consumed by a job in position p on machine m is less than or equal than the total quantity of available components. Furthermore, it ensures that the starting time of the job scheduled on machine i in position p will not be earlier than the arrival time of the component relating to the execution of this job.

Constraint (13) aims to maintain the linearity of the model. It consists to make the link between the variable *Yipt* and the starting time of job processing in position p. Constraint (14) illustrates the total completion time. Constraint (15) and constraint (16) are binary variables.

5. GENETIC ALGORITHM (GA)

5.1. Genetic Algorithm (GA)

The GA operating principle allows population evolution under different mechanisms. A population is represented by a set of N elements. The inspiration is to promote the survival and reproduction of the best adapted individuals to the environment. This adaptation is evaluated by a fitness function which is directly related to the objective function value of this individual. Operators applied to the population allow creating new individuals (crossover and mutation) and selecting the most suitable individuals (selection and replacement). Subsequently, the population is improving from one generation to the next [21]. Therefore the population is predominantly composed of adapted individuals. The essential steps of the GA are summarized as follows:

5.2. Coding

The coding of individuals consists in construct a mathematical model of the considered problem in order to study it. Afterwards, encode individuals as chromosomes. Each chromosome defines a particular configuration system. When a machine becomes available, then the next job is selected in accordance with the corresponding chromosome which will be evaluated and modified to minimize the objectif function. For example, we consider 4 jobs which must be performed on 2 machines. The chromosome is encoded in a table consisting of two lines, the first one represents jobs and the second one represents machines (Fig. **2**).

j_1	j_2	j_3	j_4
1	**2**	**1**	**1**

Fig. (2). Chromosome representation.

5.3. Selection and Replacement

The selection phase is based on the fitness function of individuals, it selects among the population; parents will generate children. This operator denotes the individuals that take part in the reproduction. Several operators are available which can be either deterministic or stochastic. In this study, the selection technique applied is elitist technique. This phase is followed by new individuals' creation step which is mainly done using crossover and mutation operators.

5.4. Crossover

The crossover operator is a stochastic operator that combines between two parents to produce two offspring called children. The crossover provides generally different children's from both parents. The technique chosen to perform this step is a two point crossover which is adapted to the structure of our parallel machines scheduling environment. To illustrate the functioning of crossover, we apply it to the previous example, so it applies as follows (Fig. **3**).

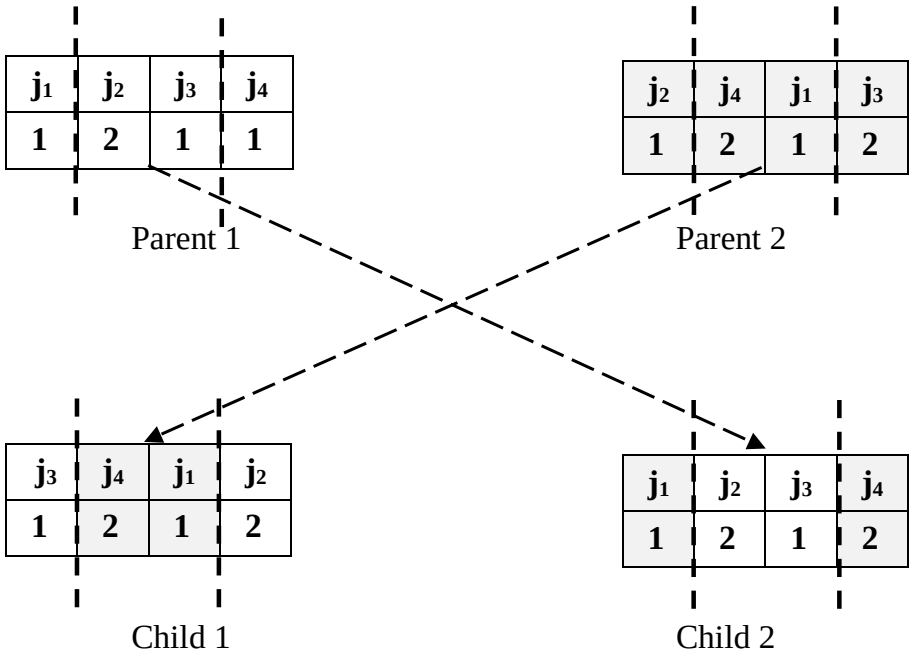

Fig. (3). Representation of two point crossover operator.

5.5. Mutation

The mutation operator is a stochastic operator that modifies an individual to create another. Following a probability, a gene of the new chromosomes is randomly changed to explore different regions of the solution space and to maintain the diversity of population through the perturbations caused in the solution, therefore, avoids falling on a local minimum. The one point mutation operator is chosen which is carried by changing assignment of jobs to machines, as shown in Fig. (**4**). The mutation is done by changing assignment of job 1.

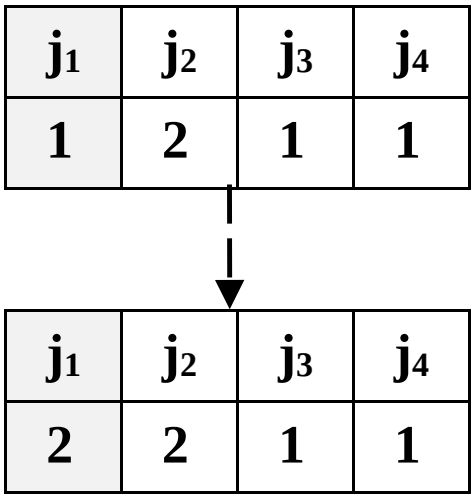

Fig. (4). Representation of one mutation operator.

5.6. Stopping Criterion

One often used criterion is the number of iterations in which no improvement is occurred consecutively, so if this criterion is reached, we save the best solution for the problem.

Therefore, the structure of the genetic algorithm can be presented as follows:

Algorithm 1: Genetic Algorithm

If there are a set of jobs not scheduled **then**
Generate a random population (initial population)
For each chromosome
Calculate the fitness of chromosome (total completion time)
Evaluate the fitness of this chromosome
If this solution is smaller than the best solution **then** update the best solution
End for
While stop criterion is not satisfied (number of iterations without improvement)
Select the individuals for the reproduction (selection)
Combination between parents to produce new individuals (crossover)
Modify individuals of the new population (mutation)
For each chromosome
Calculate the fitness of chromosome
Evaluate the fitness of this chromosome
If this solution is smaller than best solution **then** update the best solution
End for
Constitute the next generation.
End while
End if

6. HEURISTICS APPROACHES

Despite, heuristics have been extensively studied in scheduling environment; a few researchers have attempted to use these approaches in scheduling problem with consumable resources. Therefore in this section, we make a brief description of heuristics adapted to minimize the total completion time in the studied system.

6.1. Shortest Processing Time First (SPT)

This heuristic is one of well know rules in scheduling theory, the so called shortest processing time first rule. According to SPT rule, the jobs are ordered in decreasing order of their relative processing time. Thus, this heuristic retains longest jobs to the end of scheduling.

6.2. Longest Processing Time First (LPT)

Longest processing time first is a heuristic which sequences jobs in decreasing order of their processing time. This heuristic is known as the most appropriate dispatching rule for parallel machine scheduling problem. It has been used by many researchers and many authors have proposed them as good approaches to solve such problems. Consequently, LPT retains the small jobs in terms of processing time at the end of scheduling in order to equilibrate the system load.

6.3. Smallest Resource Consumption First (SRC)

Smallest Resource Consumption first (SRC) is a heuristic which consists in arranging jobs in increasing order of resources consumption, and then execute the job listing. Hence, jobs that consume few components are prioritized.

6.4. Largest Resource Consumption First (LRC)

Largest Resource Consumption first (LRC) sorts jobs in decreasing order of resources consumption, and then execute the job listing. Indeed, this technique tries to place the jobs that consume few components towards the end of the schedule.

6.5. Longest Processing-time-to-resources-consumption Ratios First (L-PT/RC)

This heuristic is suggested by Carrera *et al.* [9] to investigate the single machine problem with non-renewable resources, thereafter, in [19] we generalize this heuristic for parallel machines scheduling problem. It sequences jobs in decreasing order of their processing-time-to-resources-consumption ratios.

6.6. Shortest Processing-time-to-resources-Consumption Ratios First (S-PT/RC)

The following is inspired by the last heuristic which classifies jobs in increasing order of their processing-time-to-resources-consumption ratios.

7. COMPUTATIONAL EXPERIENCES

In this section, extensive computational experiments were conducted in order to analyze the effect of resources constraint on system behavior when the mathematical model, heuristics and the genetic algorithm are simulated. The MILP is solved on linear programming solver CPLEX. The GA and heuristics are coded in Java language and tested on a computer with Core (TM) i5 CPU 5.53 GHz and RAM 4.00 Go.

7.1. Sensitivity Analysis of the Proposed GA

In practice, the circumstances and disadvantages of metaheuristics such as their sensitivity to the parameters setting and the difficulty to ensure the solution optimality make their adaptation to achieve an effective solution a complicate task which affects the system behavior. Generally, an effective resolution of an optimization problem needs to adjust the setting of the metaheuristic applied. In this direction, the behavior of the genetic algorithm is described in terms of four parameters, which can be defined as follows:

- Mutation parameter (P_{mut})
- Crossover parameter(P_{cross})
- Number of iterations without any improvement (N_{ite})
- Population size (S_{pop})

We conducted several tests to adjust the parameters of crossover and mutation. During these tests, we varied the probability of crossover P_{cross} from the following set P_{cross}={0.7, 0.8, 0.9, 1}. Subsequently, we varied the mutation parameter P_{mut} = {0.05, 0.1, 0.15, 0.2}. Furthermore, we tried several values which belong to the following set N_{ite}={50, 100, 150, 200} in order to determine the best value of the stopping criteria. Finally, we tested several population sizes in the interval S_{pop} = {50, 100, 150, 200} to determine the best size.

The retained values of these parameters are illustrated in Table **1**.

Table 1. Retained values for the genetic algorithm configuration.

Mutation parameter	P_{mut} = 0.05
Crossover parameter	N_{ite} = 75
Number of iterations without any improvement	P_{cross} = 0.9
Population size	S_{pop} =150

7.2. Test Environment

We start this section by a brief presentation of tests instances on which we perform experiments. Note that 10 tests are performed for each instance and the component arrivals are dispersed over the working time. The remaining of this section illustrates the simulation results regarding three different experimental designs: small, medium and large size problems.

We apply some performance measures, to assess the MILP, GA and each heuristic on the system behavior. These performance indicators include:

- the number of instances solved to optimality for MILP within the 1500 sec time limit.
- the average resolution time (CPU_T) for optimally solved instances.
- the total completion time (TCT) value of each approaches to calculate the reported GAP.
- The reported GAP is calculated using TCT^{cur}-TCT^{best}-TCT^{best}, where:
- TCT^{best} is the best solution.
- TCT^{cur} is the current solution.

In addition, the variables used to perform this investigation can be summarized as follow:

N: The number of jobs belongs to:

- {4, 6, 8} for small size problems.
- {10,20, 30} for medium size problems.
- {50, 75, 100} for large size problems.

M: The number of machines belongs to:

- {2, 3} for Small Instances (SI).
- {6, 8} for Medium Instances (MI).
- {10, 15} for Large Instances (LI).

K: the number of components belongs to {2, 3, 4}.

p_j: the processing time of job *j,* is randomly generated from a uniform distribution *U* (1, 50).

Table 2. MILP performances evaluation when k=1.

Instances		MILP		GA	
M	**n**	**CPUT**	**GAP**	**CPUT**	**GAP**
2	4	1.26	0	0.14	0
	8	1.09	0	0.16	0
	12	1.25	0	0.17	0
	16	45.38	0	0.18	0
	18	>1500	/	0.19	0

Instances		MILP		GA	
m	**N**	**CPUT**	**GAP**	**CPUT**	**GAP**
3	6	1.45	0	0.15	0
	9	1.69	0	0.17	0
	12	159.4	0	0.20	0
	15	>1500	/	0.23	0

Instances		MILP		GA	
m	**n**	**CPUT**	**GAP**	**CPUT**	**GAP**
4	8	1.18	0	0.16	0
	12	1.26	0	0.19	0
	16	118.2	0	0.22	0
	20	>1500	/	0.24	0

7.3. Performance Evaluation of the MILP

In this section, we perform several tests to evaluate the MILP limits and GA performances. The execution time of a MILP is generally too important to browse completely the search space, so we have limited the CPU_T to 1500 seconds.

Table 3. MILP performances evaluation when k=2.

Instances		MILP		GA	
M	**n**	**CPU_T**	**GAP**	**CPU_T**	**GAP**
2	4	1.48	0	0.13	0
	8	11.5	0	0.15	0
	12	14.93	0	0.19	0
	16	96.9	0	0.21	0
	18	>1500	/	0.23	0

(Table 3) contd.....

Instances		MILP		GA	
m	**n**	**CPU_T**	**GAP**	**CPU_T**	**GAP**
4	8	41.5	0	0.20	0
	12	47.3	0	0.21	0
	16	>1500	/	0.22	0

Instances		MILP		GA	
m	**n**	**CPU_T**	**GAP**	**CPU_T**	**GAP**
4	8	41.5	0	0.20	0
	12	47.3	0	0.21	0
	16	>1500	/	0.22	0

Table 4. MILP performances evaluation when k=3.

Instances		MILP		GA	
M	**n**	**CPU_T**	**GAP**	**CPU_T**	**GAP**
2	4	2.1	0	0.13	0
	8	14.68	0	0.15	0
	12	155.5	0	0.18	0
	16	>1500	/	0.21	0

Instances		MILP		GA	
m	**n**	**CPU_T**	**GAP**	**CPU_T**	**GAP**
3	6	32.2	0	0.16	0
	9	65.8	0	0.18	0
	12	67.1	0	0.19	0
	15	78.3	0	0.22	0
	18	>1500	/	0.23	0

Instances		MILP		GA	
m	**n**	**CPU_T**	**GAP**	**CPU_T**	**GAP**
4	8	45.5	0	0.18	0
	12	79.5	0	0.19	0
	16	145.3	0	0.21	0
	20	>1500	/	0.23	0

Tables **2 - 5** summarize the results obtained by the GA and MILP to compute the CPU_T and the GAP for different tests problems. It can be noticed that the mathematical model is able to provide optimal solution for problems which not exceed 16 jobs, furthermore, the average computation time increases exponentially when the number of jobs increases.

Table 5. MILP performances evaluation when k=4.

Instances		MILP		GA	
M	**n**	**CPUT**	**GAP**	**CPUT**	**GAP**
2	4	2.1	0	0.14	0
	8	12.5	0	0.16	0
	12	14.8	0	0.19	0
	16	>1500	/	0.22	0

(Table 5) contd.....

Instances		MILP		GA	
m	**n**	**CPUT**	**GAP**	**CPUT**	**GAP**
3	6	41.34	0	0.17	0
	9	48.6	0	0.18	0
	12	60.1	0	0.19	0
	15	115.8	0	0.20	0
	18	>1500	/	0.22	0

Instances		MILP		GA	
m	**n**	**CPU_T**	**GAP**	**CPU_T**	**GAP**
4	8	46.44	0	0.18	0
	12	81.8	0	0.19	0
	16	185.6	0	0.25	0
	20	>1500	/	0.26	0

It can be observed also that the MILP cannot generate a feasible solution for instances greater than 16 jobs in the given time limit. This is due to the consumable resources constraint which increases the corresponding problem complexity. However, the GA gives the same results as MILP for instances solved to optimality within the allowed execution time. Moreover the average resolution time is relatively small compared to the mathematical model.

7.4. Performance Evaluation of the GA

In this section, in order to analyze the GA behavior, we will present some simulation results and their interpretations concerning the studied performances indicators.

Note that, the following constraints must be respected:

- Each machine can execute only one job at a time.
- Each job must be processed without interruption.
- All jobs should be treated to achieve the production.

7.4.1. Experimental Results for Small Size Problems

Table **6** provides the results concerning the performances indicators obtained for small size problems.

Table 6. Experimental results for small size problems.

Problem	GA		SPT		LPT		SRC		LRC		S-PT/RC		L-PT/RC	
	CPU_T	**GAP**	**CPU_T**	**GAP**	**CPU_T**	**GAP**	**CPU_T**	**GAP**	**CPU_T**	**GAP**	**CPU_T**	**GAP**	**CPU_T**	**GAP**
k=2	0.13	0.000	0.01	0.1	0.01	0.08	0.01	0.1	0.01	0.31	0.01	0.28	0.01	0.1
k=3	0.13	0.000	0.01	0.3	0.01	0.17	0.01	0.00	0.01	0.39	0.01	0.31	0.01	0.1
k=4	0.14	0.000	0.01	0.1	0.01	0.09	0.01	0.17	0.01	0.16	0.01	0.35	0.01	0.21

As we know, enhancing the performance of our system is the motivating factor for this work. Especially, the primary objective considered is the minimization of the total completion time. According to these results, the proposed optimization method based on genetic algorithm outperform the others heuristics on the basis of total completion time. Additionally, the CPU_{time} is almost similar for all heuristics but it starts to increase slowly for the genetic algorithm when the number of consumables resources increases.

It should be noted that SPT gives the worst results. Regarding heuristics that based components, SRC has almost the same behavior as LRC for small instances and surpasses it for some instances. Furthermore, heuristics based on the processing time, it is observed that LPT which is known as the most suitable sequencing rule for this problem behaves better than SPT for all cases. Finally, on the basis of the processing-time-to-resources-consumption ratios, L-PT/RC performs better than S-PT/RC. Note that the execution time is almost similar for all approximate approaches.

7.4.2. Experimental Results for Medium Size Problems

The results regarding the impact of applied heuristics on the studied performances indicators for medium size problems are illustrated in Table **7**. It can be noticed that, the execution time is the same for all rules despite an acceptable increase for genetic algorithm. It is observed also that L-PT/RC and SPT heuristics lead respectively to the best and the worst values for the studied performances indicators Moreover, it can be noted that LPT outperform SPT for all cases. On the other hand, the values concerning heuristics based resources consumption show that SRC surpasses LRC, moreover, finally L-PT/RC exceeds S-PT/RC on the basis of the processing-time-to-resources-consumption ratios. Note that the adapted metaheuristic surpass all heuristics on the basis of total completion time.

Table 7. Experimental results for medium size problems.

Problem	GA		SPT		LPT		SRC		LRC		S-PT/RC		L-PT/RC	
	CPU_T	GAP	CPU_T	GAP	CPU_T	GAP	CPU_T	GAP	CPU_T	GAP	CPU_T	GAP	CPU_T	GAP
k=2	0.39	0.000	0.01	0.31	0.01	0.01	0.01	0.19	0.01	0.18	0.01	0.23	0.01	0.01
k=3	0.45	0.000	0.01	0.34	0.01	0.04	0.01	0.21	0.01	0.31	0.01	0.35	0.01	0.02
k=4	0.52	0.000	0.01	0.41	0.01	0.05	0.01	0.24	0.01	0.42	0.01	0.41	0.01	0.01

7.4.3. Experimental Results for Large Size Problems

The results concerning the total completion time for large size problems are given in Table **8**.

Table 8. Experimental results for large size problems.

Problem	GA		SPT		LPT		SRC		LRC		S-PT/RC		L-PT/RC	
	CPU_T	GAP	CPU_T	GAP	CPU_T	GAP	CPU_T	GAP	CPU_T	GAP	CPU_T	GAP	CPU_T	GAP
k=2	1.93	0.000	0.01	0.28	0.01	0.021	0.01	0.06	0.01	0.21	0.01	0.31	0.01	0.003
k=3	2.05	0.000	0.01	0.26	0.01	0.018	0.01	0.1	0.01	0.22	0.01	0.32	0.01	0.004
k=4	2.31	0.000	0.01	0.24	0.01	0.032	0.01	0.12	0.01	0.24	0.01	0.34	0.01	0.000

It can be observed that the best solution for this problem can be obtained by genetic algorithm and it surpass all heuristics for the studied performances indicators. However, the computational time remains small for all methods. According to heuristics that are based on the processing time, it is noticed that LPT gives better results than SPT particularly when k=4. Furthermore, LRC and SRC have practically the same behavior with a small advantage to SRC. Finally, L-PT/RC outperforms S-PT/RC on the basis of the processing-time-to-resources-consumption ratios. Based on the results of this study, it would appear more interesting to focus on the utilization and exploitation of L-PT/RC heuristic compared to GA in an attempt to improve the system performances.

CONCLUSION

The scheduling problem in parallel machine environment is considered in this paper. A Mixed Integer Linear Programming model is developed that targets the optimal production scheduling in a parallel machine environment. The model takes into account all the standard constraints encountered in production scheduling as renewable resources. Furthermore, it considers special features that characterize production system, which are consumables resources. The objective is the minimization of the total completion time.

Due to the NP-hardness of the studied problem and the huge number of variables and constraints, a genetic algorithm is suggested as a suitable methodology for solving it. Simulations were performed in different size problems and the results demonstrate that genetic algorithm has the same behavior to the mathematical model for the small-size problem and it outperforms others heuristics in the basis of solution quality and computation time for large-size problem.

According to this point, it would appear more advantageous to concentrate on the application of the studied approach in the parallel machine scheduling with consumable resources in an attempt to improve system performance and more effort must be made to enrich this work which can be the subject of further studies and important implications in the following areas:

Genetic algorithm is very performing to validate the proposed study. Since, in manufacturing companies, the

decision making process is too complicated due the presence of different constraints, so achieving a good plan by applying the GA independently is quite difficult; therefore, it appears interesting to develop a local search heuristic in order to explore neighborhoods effectively.

Furthermore, the conducted research focused on minimizing total completion time in parallel machines environment. However, many objectives can be well-justified in practice, which is more realistic; therefore minimizing the tardiness is also an interesting objective. Consequently, it is noticed that a multiple objective optimization study is necessary and represents an important future direction.

Finally, production scheduling and maintenance planning are two aspects which must cooperate simultaneously in order to decrease the probability of machine breakdowns and to increase the productivity of manufacturing companies. However, despite the interdependent relationship between these two activities, they are generally planned and performed independently. Hence, it appear necessary to propose an integrated model that takes into account the maintenance and production service under non-renewable resources constraints.

CONSENT FOR PUBLICATION

Not applicable.

CONFLICT OF INTEREST

The authors declare no conflict of interest, financial or otherwise.

ACKNOWLEDGEMENTS

Declared none.

REFERENCES

[1] J. Erschler, G. Fontan, and F. Roubellat, "Encyclopédie du management Ordonnancement en ateliers spécialisés", *Helfer et Orsoni,Vuibert, Paris,* vol. 2, pp. 208-229, 1992.

[2] A. Allahverdi, C.T. Ng, T.C. Cheng, and M.Y. Kovalyov, "A survey of scheduling problems with setup times or costs", *Eur. J. Oper. Res.,* vol. 187, no. 3, pp. 985-1032, 2008.
[http://dx.doi.org/10.1016/j.ejor.2006.06.060]

[3] M.R. Garey, and D.S. Johson, *Computers and Intractability A Guide of the Theory of NP-completeness,* W.H. Freeman and company: San Francisco, 1979.

[4] J. Blazewicz, J.K. Lenstra, and A.H. Kan, "Scheduling subject to resource constraints: classification and complexity", *Discrete Appl. Math.,* vol. 5, no. 1, pp. 11-24, 1983.
[http://dx.doi.org/10.1016/0166-218X(83)90012-4]

[5] A. Toker, S. Kondakci, and N. Erkip, "Scheduling under a non-renewable resource constraint", *J. Oper. Res. Soc.,* vol. 42, no. 9, pp. 811-814, 1991.
[http://dx.doi.org/10.1057/jors.1991.152]

[6] J. Xie, "Polynomial algorithms for a single machine scheduling problems with financial constraints", *Oper. Res. Lett.,* vol. 21, pp. 39-42, 1997.
[http://dx.doi.org/10.1016/S0167-6377(97)00007-2]

[7] M. Kaspi, and D. Shabtay, "Convex resource allocation for minimizing the makespan in a single machine with job release dates", *Comput. Oper. Res.,* vol. 31, pp. 1481-1489, 2004.
[http://dx.doi.org/10.1016/S0305-0548(03)00103-5]

[8] A.A. Lazarev, and F. Werner, "Algorithms for special cases of the single machine total tardiness problem and an application to the even-odd partition problem", *Math. Comput. Model.,* vol. 49, no. 9-10, pp. 2061-2072, 2009.
[http://dx.doi.org/10.1016/j.mcm.2009.01.003]

[9] S. Carrera, M.C. Portmann, and W. Ramdane-Cherif, *Scheduling supply chain node with fixed components arrivals and two partially flexible deliveries: 5th International Conference on Management and Control of Production and Logistics*: France, 2010.
[http://dx.doi.org/10.3182/20100908-3-PT-3007.00030]

[10] R.L. Daniels, B.J. Hoopes, and J.B. Mazzola, "An analysis of heuristics for the parallel-machine flexible-resource scheduling problem", *Ann. Oper. Res.,* vol. 70, pp. 439-472, 1997.
[http://dx.doi.org/10.1023/A:1018946810121]

[11] R.L. Daniels, S.Y. Hua, and S. Webster, "Heuristics for parallel-machine flexible-resource scheduling problems with unspecified job assignment", *Comput. Oper. Res.,* vol. 26, no. 2, pp. 143-155, 1999.

[http://dx.doi.org/10.1016/S0305-0548(98)00054-9]

[12] L. Min, and W. Cheng, "A genetic algorithm for minimizing makespan in the case of scheduling identical parallel machines", *Artif. Intell. Eng.,* vol. 13, pp. 399-403, 1999.
[http://dx.doi.org/10.1016/S0954-1810(99)00021-7]

[13] S.F. Serifoglu, and G. Ulusoy, "Parallel machine scheduling with earliness and tardiness penalties", *Comput. Oper. Res.,* vol. 26, no. 8, pp. 773-787, 1999.
[http://dx.doi.org/10.1016/S0305-0548(98)00090-2]

[14] A.S. Mendes, F.M. Muller, P.M. França, and P. Moscato, "Comparing meta-heuristic approaches for parallel machine scheduling problems", *Prod. Plann. Contr.,* vol. 13, pp. 143-154, 2002.
[http://dx.doi.org/10.1080/09537280110069649]

[15] J.W. Fowler, S.M. Horng, and J.K. Cochran, "A hybridized genetic algorithm to solve parallel machine scheduling problems with sequence dependent setups", *Int. J. Ind. Eng.: Theory Appl. Pract.,* vol. 10, pp. 232-243, 2003.

[16] A.D. Wilson, R.E. King, and T.J. Hodgson, "Scheduling non-similar groups on a flow line: Multiple group setups", *Robot. Comput.-Integr. Manuf.,* vol. 20, pp. 505-515, 2004.
[http://dx.doi.org/10.1016/j.rcim.2004.07.002]

[17] K. Li, Y. Shi, S.L. Yang, and B.Y. Cheng, "Parallel machine scheduling problem to minimize the makespan with resource dependent processing times", *Appl. Soft Comput.,* vol. 11, no. 8, pp. 5551-5557, 2011.
[http://dx.doi.org/10.1016/j.asoc.2011.05.005]

[18] K. Lee, L. Lei, M. Pinedo, and S. Wang, "Operations scheduling with multiple resources and transportation considerations", *Int. J. Prod. Res.,* vol. 51, no. 23-24, pp. 7071-7090, 2013.
[http://dx.doi.org/10.1080/00207543.2013.781283]

[19] F. Belkaid, Z. Sari, and M. Souier, "A genetic algorithm for the parallel machine scheduling problem with consumable resources", *Int. J. Appl. Metaheuristic Comput.,* vol. 4, no. 2, pp. 17-30, 2013.
[http://dx.doi.org/10.4018/jamc.2013040102]

[20] F. Belkaid, F. Yalaoui, and Z. Sari, *A hybrid genetic algorithm for parallel machine scheduling problem with consumable resources,* IEEE International Conference on Control Decision and Information Technologies CoDIT'13, 2013.
[http://dx.doi.org/10.1109/CoDIT.2013.6689534]

[21] P.C. Chang, S.H. Chen, and K.L. Lin, "Two-phase sub population genetic algorithm for parallel machine-scheduling problem", *Expert Syst. Appl.,* vol. 29, pp. 705-712, 2005.
[http://dx.doi.org/10.1016/j.eswa.2005.04.033]