Management and Production Engineering Review
Volume 14 • Number 3 • September 2023 • pp. 44–56
DOI: 10.24425/mper.2023.147189

Optimization of Job Shop Scheduling Problem by Genetic Algorithms: Case Study

Habbadi SAHAR[1], Brahim HERROU[1], Souhail SEKKAT[2]

[1] *Sidi Mohamed Ben Abdellah University, Faculté des Sciences Techniques de Fès, Industrial Engineering Department, Morocco*
[2] *Ecole Nationale Supérieure d'Arts et Métiers ENSAM MEKNES, Industrial Engineering Department, Morocco*

Received:15 December 2022
Accepted: 20 April 2023

Abstract
The Job Shop scheduling problem is widely used in industry and has been the subject of study by several researchers with the aim of optimizing work sequences. This case study provides an overview of genetic algorithms, which have great potential for solving this type of combinatorial problem. The method will be applied manually during this study to understand the procedure and process of executing programs based on genetic algorithms. This problem requires strong decision analysis throughout the process due to the numerous choices and allocations of jobs to machines at specific times, in a specific order, and over a given duration. This operation is carried out at the operational level, and research must find an intelligent method to identify the best and most optimal combination. This article presents genetic algorithms in detail to explain their usage and to understand the compilation method of an intelligent program based on genetic algorithms. By the end of the article, the genetic algorithm method will have proven its performance in the search for the optimal solution to achieve the most optimal job sequence scenario.

Keywords
Optimization; Metaheuristics; Scheduling; Job Shop Scheduling problem; Genetic Algorithms; Simulation.

Introduction

The present article will address a widespread problem in the industrial sector, and its solution will enable industry professionals to withstand and thrive in their field of activity. The problem concerns the optimization of a set of production processes in terms of time, quality, and deadlines.

Firstly, this article will provide a detailed explanation of genetic algorithms. Then, it will explore the job shop scheduling problem, which involves a set of jobs that are completed on machines through operations that can have multiple scenarios for completion. The most optimal scenario is equivalent to the minimum makespan.

The objective of this article is to minimize the makespan of the job shop scheduling problem using the genetic algorithm method. Genetic algorithms are based on the principle of natural phenomena, where the process starts with an initial population (in the case of the job shop scheduling problem, the initial population represents the first iteration or the series of operations among several other iteration configurations), then seeks out individuals with a higher chance of survival and reproduction through the Gantt chart, integrates them into the population, and selects two parents to cross and mutate their individuals to generate two child, who will be the future parents. The process repeats in the same manner until the iteration generates the most optimal configuration.

The Job Shop Scheduling (JSS) problem dealt with in the rest of this article will consider the following elements: a set of 6 jobs carried out on the 6 machines available to manufacture products in a minimum time while optimizing the sequences of the job.

The problem consists of searching for the most optimal iteration that represents the sequence configuration of operations using an algorithm that employs an evolutionary method known as genetic algorithms. The goal is to generate a model that can be applied in any situation, with the objective of generating the minimal possible makespan.

***Corresponding author**: Habbadi Sahar – Sidi Mohamed Ben Abdellah University, Faculté des Sciences Techniques de Fès, Industrial Engineering Department, Morocco, e-mail: habbadis-ahar@gmail.com*

The rest of this paper is structured as follows: In Section 2, literature review providing an overview of different articles that have studied the same subject using various methods. In Section 3, the steps involved in genetic algorithms in a general manner. In Section 4, the steps of theoretical algorithm, then the case study in Section 5 and solution methodologies are discussed in Section 6 by giving the details of each step of the algorithm. Finally Section 7, provides results, conclusion and some research directions to be investigated in future research work in Section 8.

Literature review

The job shop scheduling problem is a widely studied optimization problem, in which multiple jobs are processed on several machines. Each job consists of a sequence of tasks, which must be performed in a given order, and each task must be processed on a specific machine. In this section we will first perform a literature review about algorithms resolving Job shop scheduling problem.

The job shop scheduling problem has been tackled by several researchers, including Asadzadeh & Zamanifa (2010) in their article "An agent-based parallel approach for the job shop scheduling problem with genetic algorithms". The authors describe a method to solve the production scheduling problem in production workshops, also known as the job shop problem. This problem involves determining the order in which tasks should be performed on different machines while respecting time and resource constraints.

The authors propose an approach based on genetic algorithms, which is a metaheuristic search technique inspired by the theory of evolution. They also use an agent-based approach, where each task is represented by an autonomous agent that can communicate with other agents to exchange information and collaborate on problem-solving.

The proposed approach is designed to be parallel, meaning that multiple processes can be executed simultaneously to speed up the search for the optimal solution. The authors tested their approach on several instances of the job shop problem and found that it yielded better results than other conventional solving methods.

In the article "Hybrid genetic algorithm with variable neighborhood search for flexible job shop scheduling problem in a machining system," authors Sun et al. (2023) present a solution method for the flexible job shop scheduling problem in a machining system.

The authors propose a hybrid algorithm that combines a genetic algorithm and variable neighborhood search to improve solution efficiency. The algorithm uses genetic operators to explore the search space and neighborhood search strategies to improve existing solutions.

Experimental results show that the proposed hybrid algorithm is capable of providing high-quality solutions and significantly reducing processing time for large-scale problem instances. The authors conclude that their algorithm is effective in solving the flexible job shop scheduling problem in a machining system.

In summary, the experimental results suggest that the proposed hybrid algorithm is effective in solving the flexible job shop scheduling problem in a machining system, providing high-quality solutions with improved processing efficiency. The authors also show that the use of variable neighborhood search in their hybrid algorithm improves the efficiency of the search space exploration, enabling faster and better-quality solutions.

The article "Neural network and genetic algorithm-based hybrid approach to expanded job-shop scheduling" authors Yu & Liang (2001), propose a hybrid resolution approach for the extended job-shop scheduling problem. The authors combined an artificial neural network and a genetic algorithm to improve the efficiency and quality of scheduling solutions. The neural network is used to predict the processing times of tasks, while the genetic algorithm is used to optimize the assignment of tasks to machines.

Experimental results show that the proposed hybrid approach is capable of providing high-quality solutions for large-scale problem instances, particularly for cases where task processing times are uncertain and variable. The authors also demonstrated that the hybrid approach is more effective than traditional methods for solving this problem.

By using a test dataset to evaluate the performance of their approach, the authors showed that their hybrid approach improved the efficiency and quality of solutions compared to traditional methods, especially for cases with a large number of machines and tasks.

In summary, the proposed hybrid approach in this article, which combines a neural network and a genetic algorithm, is effective in solving the extended job-shop scheduling problem by providing high-quality solutions with improved processing efficiency.

Vilcot & Billaut (2008) in their article "A tabu search and a genetic algorithm for solving a bicriteria general job shop scheduling problem" they describe two metaheuristic algorithms, namely tabu search and genetic algorithm, for solving a production scheduling problem in a job shop environment. The problem involves finding a sequence of operations for each production task that minimizes two performance criteria:

total production time and task tardiness. The problem is NP-hard, meaning that it is very difficult to find an optimal solution in a reasonable amount of time.

To solve this problem, the authors developed two metaheuristic algorithms: a tabu search algorithm and a genetic algorithm. Both algorithms are based on an iterative approach that starts with an initial solution and improves this solution at each iteration.

The tabu search algorithm is designed to explore the solution space by avoiding solutions that have already been explored. It uses a tabu list to record recently visited solutions and to avoid revisiting these solutions. The algorithm explores the search space by modifying the sequence of operations for each production task.

The genetic algorithm is based on a natural selection and evolution approach. It uses crossover and mutation operators to create new solutions by combining existing solutions. Solutions are evaluated using a multicriteria objective function that takes into account the two performance criteria.

Both algorithms were tested on a set of increasingly sized production scheduling problems. The results showed that both algorithms were able to find good solutions in a reasonable amount of time. The genetic algorithm generally produced better solutions than the tabu search algorithm, but it was also slower due to the high number of operations required to create new solutions.

The article titled "Hybrid genetic algorithms for minimizing makespan in dynamic job shop scheduling problem", authors Kundakcı & Kula (2016) present a hybrid approach to solve the scheduling problem in a dynamic workshop. This problem is known to be NP-hard. The authors propose a hybrid approach that combines a genetic algorithm (GA) with a local search (LS) method. The GA is used to explore the search space, while the LS is used to improve the quality of solutions obtained. This approach yields superior quality solutions while reducing computation time.

The scheduling problem in a dynamic workshop is defined as follows: there is a set of tasks to be performed, each of which must be processed on a specific machine. Each task has a start and end date, as well as a specific execution time. The objective is to minimize the makespan, which is the time required to complete all tasks.

The authors also propose a specific mutation operator to address the dynamic nature of the scheduling problem. This operator modifies the task execution order to adapt to changes in time constraints.

Experimental results show that the proposed hybrid approach has better performance in terms of solution quality and computation time compared to traditional methods such as standard GAs and LS. The authors also conducted sensitivity tests to evaluate the impact of different algorithm parameters on performance.

In conclusion, the proposed hybrid approach can be useful for solving complex real-time scheduling problems with dynamic data. Experimental results show that this approach yields superior quality solutions while reducing computation time.

In the article by Shen et al. (2021), the two-phase genetic algorithm (CTGA) is applied to the case of a generalized flexible flow shop problem (GFFS) while introducing mixed-integer multi-objective mixed-integer programming (MIP) for the GFFS, then the observation that the CTGA is efficient for the multi-objective optimization due to the flexibility provided by the Genetic algorithm.

The work of Báez et al. (2019) have the same objective as our article, it aims to study a machine scheduling problem and the criterion used to assess the quality of the planning is the makespan, except that the method used to solve this problem is based on hybrid algorithms which combines GRASP and variable neighborhood search metaheuristics unlike our work which will mainly focus on genetic algorithms.

Chien & Lan (2021) discuss the case of dynamic planning for semiconductor manufacturing while developing an agent-based approach that integrates deep reinforcement learning and hybrid genetic algorithm for independent parallel machine planning.

Swan et al. (2022) talk in their article about metaheuristic which is based on an iterative process that guides and modifies the operations of subordinate heuristics to efficiently produce high quality solutions. At each iteration, it manipulates either a unique complete (or partial) solution, or a set of such solutions.

The Job Shop scheduling problem is one of the most difficult combinatorial optimization problems where research has not yet been able to develop a performing algorithm to generate a satisfying optimal sequence.

Wein & Chevalier (1992) define a job shop scheduling problem with three dynamic decisions: assign due dates to jobs arriving exogenously, release jobs from a backlog to the job shop, and sequence jobs to each of two shifts workshop work.

Several researchers have worked on this subject while using different methods including:

The use of Lagrangian relaxation to schedule job shops Wein & Chevalier (1992) offer a general two-step approach to solving this problem: release and sequence work to minimize inventory of work in progress, subject to completion of work at a specified rate, and given the policies of set deadlines dates that attempt to minimize the lead time to the due date, subject to the late work constraint.

Cheng et al. (1996) discuss in their article: A tutorial survey of job-shop scheduling problems using genetic algorithms—I the representation on genetic algorithms and the different representation schemes proposed for JSP in part I of his article and various hybrid approaches of genetic algorithms and conventional heuristics in the second part.

Ritwik & Deb (2011) in their paper: A genetic algorithm-based approach for optimization of scheduling in job shop environment develops a genetic algorithm based approach to solve the scheduling optimization problem in Job Shop manufacturing environment.

The following table collects all the articles described above while classifying them by date and method (Table 1).

According to our analysis, it has been found that recently the handling of challenging issues is being addressed by the majority of researchers using artificial intelligence methods, particularly through metaheuristic, agent based approaches, neural networks, hybrid algorithms. These methods have demonstrated high efficiency and performance in terms of the generated results especially, the hybrid algorithm has demonstrated its effectiveness in rapidly solving NP-hard problems with an optimal solution.

Table 1
Classification of articles and methods used by year

Article	Year	Field
A Broader View of the Job-Shop Scheduling Problem (Wein et al., 1992)	1992	Dynamic decisions
A practical approach to job-shop scheduling problems (Hoitomt, et al., 1993)	1993	Lagrangian relaxation
A tutorial Survey of job-shop scheduling problems (Cheng et al., 1996)	1996	Genetic algorithms and conventional heuristics
Neural network and genetic algorithm-based hybrid approach to expanded job-shop scheduling (Yu & Liang, 2001)	2001	Artificial neural network and a genetic algorithm
A tabu search and a genetic algorithm for solving a bicriteria general job shop scheduling problem (Vilcot & Billaut, 2008)	2008	Tabu search and genetic algorithm
An agent-based parallel approach for the job shop scheduling problem (Asadzadeh & Zamanifar, 2010)	2010	Genetic algorithms Agent-based approach
A genetic algorithm-based approach for optimization of scheduling in job shop environment (Ritwik & Deb, 2019)	2011	Based approach
A hybrid metaheuristic algorithm for a parallel machine scheduling problem with dependent setup times (Báez et al., 2019)	2019	Hybrid algorithms
Hybrid genetic algorithms for minimizing makespan in dynamic job shop scheduling problem (Kundakcı & Kulak, 2016)	2016	Hybrid approach
Agent-based approach integrating deep reinforcement learning and hybrid genetic algorithm for dynamic scheduling for Industry 3.5 smart production (Chien & Lan, 2021)	2021	Agent-based approach: deep reinforcement learning and hybrid genetic algorithm
A parallel genetic algorithm for multi-objective flexible flowshop scheduling in pasta manufacturing. Computers & Industrial Engineering (Shen et al., 2021)	2021	Genetic algorithm
Metaheuristics "In the Large" (Swan et al., 2022)	2022	Metaheuristic
Hybrid genetic algorithm with variable neighborhood search for flexible job shop scheduling problem in a machining system (Sun et al., 2023)	2023	Hybrid algorithm that combines a genetic algorithm and variable neighborhood

Genetic algorithm steps

Initialization: The first step in a genetic algorithm is to create an initial population of potential solutions to the problem at hand. Each solution is represented as a string of bits or chromosomes.

Fitness Evaluation: The fitness of each solution in the population is evaluated using an objective function that measures how well each solution solves the problem. The objective function should be carefully designed to ensure that the best solutions receive a higher fitness score.

Selection: After evaluating the fitness of each solution, the genetic algorithm selects the best solutions to be used as parents for the next generation. This is usually done using a selection method, such as roulette wheel selection, tournament selection, uniform selection or rank selection.

Crossover: The selected parents are combined using a crossover operator to create new solutions for the next generation. The crossover operator takes two parent solutions and combines them to create one or more new child solutions.

Mutation: To introduce new genetic material into the population, the genetic algorithm applies a mutation operator to some of the new child solutions. The mutation operator randomly modifies one or more genes in the solution to create a new solution that has not been seen before.

Termination: The genetic algorithm continues to create new generations of solutions until a termination criterion is met. The termination criterion could be a maximum number of generations, a minimum fitness score, or a convergence of the population to a specific solution.

Solution: When the genetic algorithm terminates, the best solution found during the entire process is returned as the final output.

Theoretical algorithms

Initial population generation:

a. Define the number of chromosomes (potential solutions) in the initial population, denoted as N_pop.
b. Each chromosome represents a job scheduling sequence. Each job is represented by a letter, and the scheduling sequence is a string of these letters. For example, if we have 3 jobs to schedule, chromosomes could be "O11/O14/O15", "O14/O11/O15", "O15/O14/O11", "O15/O11/O14", "O11/O15/O14", "O14/O15/O11".
c. Generate N_pop chromosomes randomly.

Evaluation of the initial population:

a. For each chromosome in the population, calculate fitness function which corresponds to the total processing time (makespan) using the corresponding scheduling sequence;

While (fitness value! = termination criteria);

{

a. Selection: choose parent chromosomes for the next generation using a Uniform selection (for our case);
b. Crossover: apply a crossover operator to create new chromosomes;
c. Mutation: apply a mutation operator to each new chromosome with a certain probability. A common mutation operator is inversion, where a random section of a chromosome is reversed;
d. Evaluation: evaluate the new chromosomes by calculating the corresponding makespan (fitness function);
e. Replacement: replace the least performing chromosomes of the current population with the newly generated chromosomes;
f. Check stopping criteria: if a stopping criterion is met (e.g., a maximum number of iterations is reached), stop the algorithm and return the best solution found. Otherwise, return to step a;

}

Return the best solution found;

Job shop scheduling problem

"Scheduling", is a research field that aims to determine the order and timing of tasks to be executed while respecting a set of constraints. In the context of the "job shop scheduling problem", there are two important constraints: resource constraints and precedence constraints.

Resource constraints refer to the limited availability of certain equipment or machines necessary to perform certain tasks. In a manufacturing workshop, for example, some machines can only process one task at a time or require preparation time before they can be used. It is therefore important to plan the use of these resources to avoid scheduling conflicts.

Precedence constraints, on the other hand, refer to the dependency relationships between tasks. Some tasks can only be performed once others have been completed, or must be performed in a certain order for the process to work correctly. In the context of the "job shop scheduling problem", each task must be performed on a specific machine, and some tasks can

only be performed after others have been completed on the same machine.

The job shop scheduling problem consists of finding an optimal sequence of tasks to be executed on each machine while respecting resource and precedence constraints. The goal is to minimize the total time required to complete all tasks. This problem is NP-complete, which means that there is probably no exact and efficient algorithm to solve large instances. However, many heuristics and approximation algorithms have been proposed to find acceptable solutions within reasonable timeframes.

In their article, Ghasemi et al. (2021) state that there are various parameters that affect production, particularly in the case of products that require advanced decision-making. For instance, an increase in production speed or load can lead to difficulties in the decision-making process. This problem has led to a line of research for artificial intelligence researchers who are interested in developing decision support tools. Planning is also a critical component of the decision-making process. Therefore, it is essential to develop an algorithm that can seek the optimal solution to resolve the original problem.

Regarding Tamssaouet et al. (2022), they find it challenging to optimize a global scheduling model due to the difficulties that arise in scheduling work areas locally while ensuring coordination between local schedules.

The analysis of JSS problems provides valuable insights into solving scheduling problems encountered in complex and realistic systems. Therefore, heuristics and metaheuristics are preferred methods for job shop scheduling. The Genetic Algorithm is considered the most well-known optimization technique for a class of combinatorial problems.

In this article we consider a matrix of order 6×6 which corresponds to 6 Jobs and 6 machines.

– The machines are subjected to the operations below:

It is supposed that:

- The machine setup time is 0 for all machines,
- Each machine is a unique and we have only one machine from each type,
- The execution time of the operations is the same on each machine,
- For each job, the sequence of operations is predetermined,
- Each machine executes only one operation at a time,
- The order of the machine is fixed as shown in Table 2, because what really influences the makespan is the job sequences.

Table 2
Distribution of operation on machines

	M_1	M_2	M_3	M_4	M_5	M_6
J_1	O11	O21			O51	
J_2		O22	O32			
J_3			O33			O63
J_4	O14		O34	O44	O54	
J_5	O15	O25		O45		O65
J_6			O36	O46		O66

The following symbols are defined:

$J = \{J_1, J_2, J_3, \ldots, J_j, \ldots, J_n\}$ represents the job set.

$M = \{M_1, M_2, M_3, \ldots, M_i, \ldots, M_m\}$ represents the machine set.

$Oij \rightarrow$ O: Operation, i: corresponds to the machine number and j: to the job number.

The first step to resolve a combinatorial problem by genetic algorithm is to establish population initial.

According to Loukil et al. (2005) population initialization is an important task in evolutionary algorithms because it can affect both the convergence speed and the quality of the final solution. If no information about the solution is available, then random initialization is the most commonly used method to generate candidate solutions.

There is two ways to construct the initial population which are:

- Random Initialization:
  Initial population contains only random solutions.
- Heuristic Initialization:
  Initial population contains solutions founded by heuristic method.

$\rightarrow$ In this article the random initialization is preferred as a method in order to avoid premature convergence.

- The coding of the genetic algorithm is also part of the first step that is carried out to create our population and it is a crucial step to succeed in finding the optimal solution.
- The coding step consists on representing a solution as a string that conveys the necessary information.
- It consists also in modeling each solution by a chromosome, we consider:
  $P = O11, O14, O15, O22, O21, O25, O33, O32, O34, O36, O44, O45, O46, O51, O54, O63, O65, O66.$

The initial population corresponds to the set of feasible solutions to solve this problem.

The priority order or what we call the sequence of operations that we randomly chose at the level

of the first generation representing the initial population, Figure 1 illustrates the random choice of the first iteration.

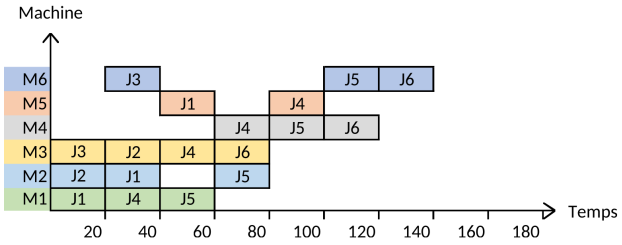

Fig. 1. Gantt chart of the first iteration

After selecting the initial population, a Gantt chart is developed too, on one hand, fix the order of operations with the same sequence to avoid violating resource constraints and, on the other hand, determine the value of the fitness function that represents the makespan.

Table 3 of sequences matrix represents the set of feasible solutions in initial population iteration in order to lead the program to an optimal sequences in the following steps.

Table 3
Table of sequences matrix

	M_1	M_2	M_3	M_4	M_5	M_6
J_1	1	2			1	
J_2		1	2			
J_3			1			1
J_4	2		3	1	2	
J_5	3	3		2		2
J_6			4	3		3

The order of job operations is represented by random sequence numbers. The objective is to run a program based on an evolutionary method to generate multiple iterations that will subsequently modify this initial order, in order to search for the most optimal order that will lead to the minimum makespan.

In Table 3: the red number 1 corresponds to the first operation order of job 1 in machine 1 and in machine 5, meaning that the job can be carried out in the first place either in machine 1 or in machine 5. However, there are multiple scenarios or iterations to consider.

The initial population is chosen randomly, but it must be feasible. Therefore, it is necessary to assign only one operation to each machine and maintain the order. The algorithm must account for this condition to prevent overlapping in the execution of certain operations. As shown in the matrix sequence table, the sequence of the job 1 can be executed on either machine 1 or machine 5 simultaneously, which is not permissible in the programming. Similarly, job 3 can be executed on either M_3 or M_6, and job 4 can be executed on M_1 or M_5, etc. (Table 2).

The number of iteration in this case of each machine is equal to:

$$N = (N_{mi}!), \quad i = \{1, 2, 3, 4, 5, 6\}$$

with: N_{mi} – the number of iteration of job on each machine mi.

Therefore, there are 50 possible iterations between jobs for each machine.

Let's defining the priority of job execution on machines:

Let's consider the bellow order of sequences: M_1, M_2, M_3, M_4, M_5, and M_6 (the order of machines is fixed).

To evaluate each chromosome, it is necessary to calculate the weight of each chromosome. The value of makespan can use Gantt diagram.

Problem statement

a. Population initialization

The initial population is formed based on the Table 2 that illustrates the execution of operations for each job on each machine.

The initial population corresponds to the set of feasible solutions to solve this problem.

P = O11, O14, O15, O22, O21, O25, O33, O32, O34, O36, O44, O45, O46, O51, O54, O63, O65, O66.

b. Fitness function

The fitness function measures the sum of the processing times for each task, taking into account the machine changeover times between tasks.

Thus, the goal of JSSP optimization is to minimize the fitness function, which means finding the task schedule that minimizes the total time required to complete all tasks.

$$C_{\max} = \max(C_1, C_2, \ldots, C_n)$$

where, C_j is the completion time of job j which corresponds to the release time of job j.

c. Selection

This step involves selecting the parents:

We chose uniform selection to avoid premature convergence and to respect the natural reproduction phenomenon that occurs randomly.

Cohan (1984) says that Uniform selection is generally supposed to cause convergence between populations while drift has the effect of causing divergence.
→ We select parents from the existing population by randomly choosing two values of the fitness function to define the new parents.

The following parents can be considered as two feasible solutions:

$$P1 = [O11, O14, O15, O22, O21, O25, O33, O32, O34, O36, O44, O45, O46, O51, O54, O63, O65, O66]$$

$$P2 = [O11, O15, O14, O22, O21, O25, O33, O32, O36, O34, O45, O44, O46, O51, O54, O63, O65, O66]$$

d. Crossover

The OX (Order Crossover) method works by selecting two random cut points on the parent sequences. The genes between the two cut points are preserved in the order they appear in the first parent. Then, the remaining genes are copied in order from the second parent, avoiding copying any gene already present in the first part of the child sequence. Finally, any missing genes are inserted into the child sequence in the order they appear in the second parent. This method was first introduced by Davis in 1985.

This step involves crossing two parents to create two children or offspring which refers to the new individuals created by applying genetic operators (OS1 and OS2).

We will use the Order-based Crossover (OX) method.

Here is a step-by-step explanation of the OX crossover:

1. Select two parent individuals from the population.
2. Choose a random range or section within the parent chromosomes.
3. Copy the selected section from the first parent to the corresponding positions in both offspring.
4. Starting from the end of the selected section in the first parent, iterate through the second parent, and for each element that is not already present in the first offspring, add it to the offspring in the order it appears.
5. Repeat step 4 for the second offspring, starting from the end of the selected section in the second parent.

The crossover of the first iteration of our problem is illustrated in Figure 2.

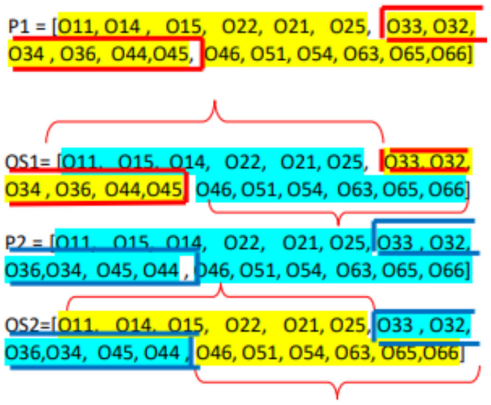

Fig. 2. Crossover of the first iteration

e. Mutation

Concerning the OS1 (we chose not to apply the mutation). Below the Gantt chart of the new job sequences (Figure 3):

OS1 = [O11, O15, O14, O22, O21, O25, O33, O32, O34, O36, O44, O45, O46, O51, O54, O63, O65, O66]

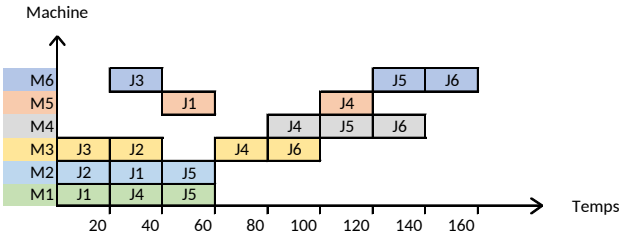

Fig. 3. Gantt chart after mutation

Fitness function: T = 160 seconds

Mutation by exchange at the OS2 (Figure 4)

O11 ← O15

O15 ← O11

OS2 = [O15, O14, O11, O22, O21, O25, O33, O32, O36, O34, O45, O44, O46, O51, O54, O63, O65, O66]

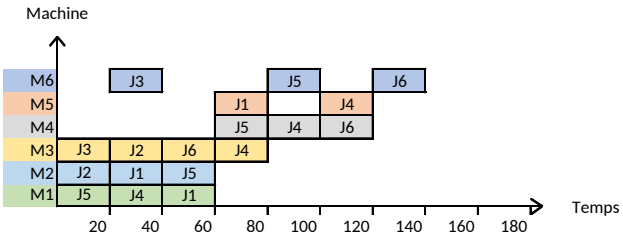

Fig. 4. Gantt chart after mutation

Fitness function: T = 140 seconds

The new population:

P1 = [O11, O15, O14, O22, O21, O25, O33, O32, O34, O36, O44, O45, O46, O51, O54, O63, O65, O66]

P2 = [O15, O14, O11, O22, O21, O25, O33, O32, O36, O34, O45, O44, O46, O51, O54, O63, O65,O66]

The fitness function for OS2 has the lowest value. However, we will select two offspring to be the future parents in the new population as we have chosen to use uniform selection in the selection step.

The new population (second iteration):

Where:

P1 = [O11, O15, O14, O22, O21, O25, O33, O32, O34, O36, O44, O45, O46, O51, O54, O63, O65, O66]

P2 = [O15, O14, O11, O22, O21, O25, O33, O32, O36, O34, O45, O44, O46, O51, O54, O63, O65,O66]

The cross between the two parents P1 and P2 is shown in Figure 5.

Crossover:

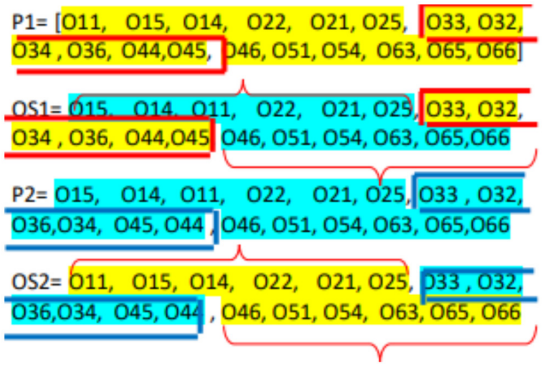

Fig. 5. Crossover of the second iteration

Mutation:

→ Mutation by exchange at the OS1:

O21 ← O25

O44 ← O45

OS1 = O15, O14, O11, O22, O25, O21, O33, O32, O34, O36, O45, O44, O46, O51, O54, O63, O65, O66

Below the Gantt chart of the new job sequences after mutation at the OS1 (Figure 6)

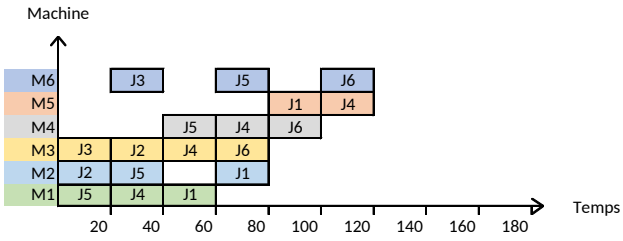

Fig. 6. Gantt chart after mutation

T = 120 seconds

→ Mutation by exchange at the OS2: O44 ← O46

OS2 = O11, O15, O14, O22, O21, O25, O33, O32, O36, O34, O45, O46, O44, O51, O54, O63, O65, O66

Below the Gantt chart of the new job sequences after mutation at the OS2 (Figure 7):

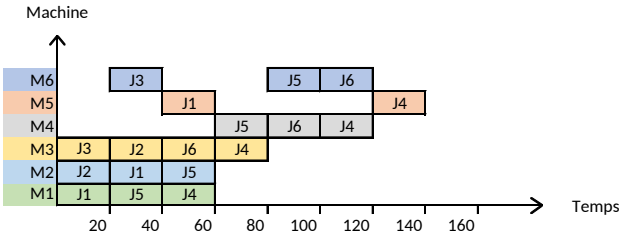

Fig. 7. Gantt chart after mutation

T = 140 seconds

The details of the iterations are in the appendix. We skip directly to iteration number 7 where the result appeared.

The new population (7^{th} iteration) is:

P1 = O14, O15, O11, O22, O25, O21, O36, O34, O33, O32, O46,O44, O45, O54, O51, O63, O66, O65

P2 = O14, O15, O11, O22, O25, O21, O33, O32, O36,O34, O45, O46, O45, O51, O54, O63, O66, O65

The cross between the two parents P1 and P2 is shown in Figure 8.

Crossover:

Fig. 8. Crossover of the seventh iteration

Mutation: At the OS1 (Figure 9):

O63 ← O66

O51 ← O54

P1 = O14, O15, O11, O22, O25, O21, O36, O34, O33, O32, O46, O44, O45, O54, O51, O66, O63, O65

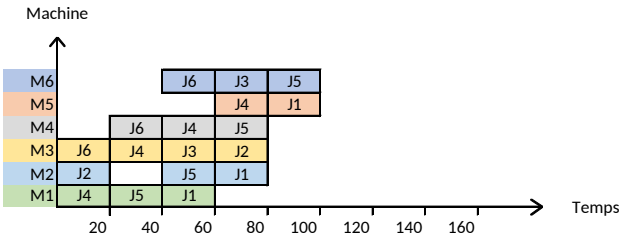

Fig. 9. Gantt chart

T = 100 seconds

Mutation: At the OS2 (Figure 10):

O32 ← O36

O45 ← O46

O54 ← O51

P2 = O14, O15, O11, O22, O25, O21, O33, O36, O32, O34, O46, O45, O44, O51, O54, O63, O66, O65

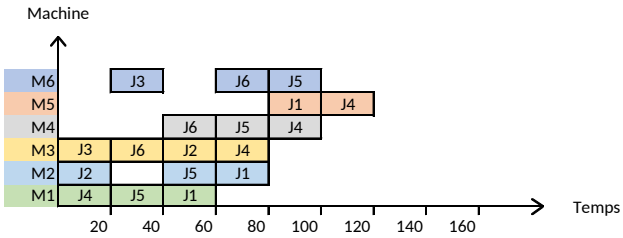

Fig. 10. Gantt chartafter mutation at the OS2

The best fitness function value corresponds to this scenario:

Pc = O14, O15, O11, O22, O25, O21, O36, O34, O33, O32, O46, O44, O45, O54, O51, O66, O63, O65

with: Fitness function T = 100 seconds, where: Pc: optimal scenario.

The research space contains 50 iterations. We manually proceeded with 7 iterations and obtained the best solution. The next step is to develop an algorithm and program that can automatically generate the optimal solution.

Results and discussion

The fitness function value of the new parents is minimal.

Genetic algorithms produce high-quality results that optimize problems classified as NP difficult.

These algorithms rely heavily on randomness; starting the iteration process with random methods enables us to explore a larger range of solutions in the search space and increases our chances of finding the global optimal solution.

The limitations of the genetic algorithm method are mainly evident in the size of the search space: The number of possible solutions for a job shop problem can be very large, making it difficult for genetic algorithms to search for the optimal solution. Genetic algorithms may require a lot of computation time to explore the entire search space as well as in determining the parameters: Genetic algorithms require the determination of several parameters such as population size, mutation and crossover probability, selection method, etc. Determining these parameters can be difficult and can have a significant impact on the quality of the solution obtained.

This case study provides a detailed explanation of the functioning of genetic algorithms and their approach to exploring a search space through iterations in order to find the most optimal scheduling in a job shop scheduling problem, allowing for a reduction in total manufacturing time.

Conclusion and perspective

The present case study provided a detailed explanation of the functioning of genetic algorithms in the job shop scheduling problem, ultimately leading to the desired result of finding the most optimal production time, allowing for the following achievements:

- After completing only seven iterations, the program was able to find the most optimal population that represents the minimal makespan.
- The search time for the optimal solution is optimized.

The next step is to develop an algorithm and program that can automatically generate the optimal solution.

Annex

The new population (third iteration):

P1 = O15, O14, O11, O22, O25, O21, O33, O32, O34, O36, O45, O44, O46, O51, O54, O63, O65, O66

P2 = O11, O15, O14, O22, O21, O25, O33, O32, O36, O34, O45, O46, O44, O51, O54, O63, O65, O66

The third iteration is:

The cross between the two parents P1 and P2 is shown in Figure 11.

Crossover:

Fig. 11. Crossover of the third iteration

Mutation:
→ Mutation by exchange at the OS1 (Figure 12):
O15 ← O14
O45 ← O44
P1 = O11, O14, O15, O22, O21, O25, O33, O32, O34, O36, O44, O45, O46, O51, O54, O63, O65, O66

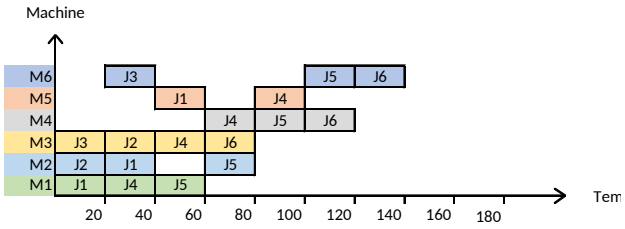

Fig. 12. Gantt chart after mutation

T = 140 seconds
→ Concerning OS2 (we chose not to apply the mutation (Figure 13)):
P2 = O15, O14, O11, O22, O25, O21, O33, O32, O36, O34, O45, O46, O44, O51, O54, O63, O65, O66

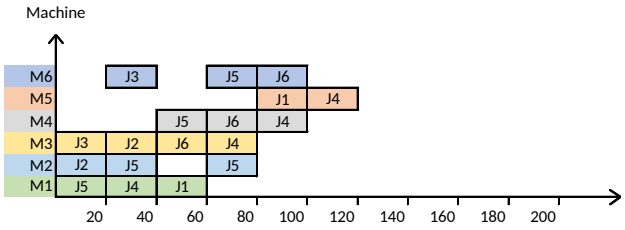

Fig. 13. Gantt chart after mutation at the OS2

T = 120 seconds
The new population (fourth iteration) is as follows:
P1 = O11, O14, O15, O22, O21, O25, O33, O32, O34, O36, O44, O45, O46, O51, O54, O63, O65, O66
P2 = O15, O14, O11, O22, O25, O21, O33, O32, O36, O34, O45, O46, O44, O51, O54, O63, O65, O66
The cross between the two parents P1 and P2 is shown in Figure 14

Crossover:

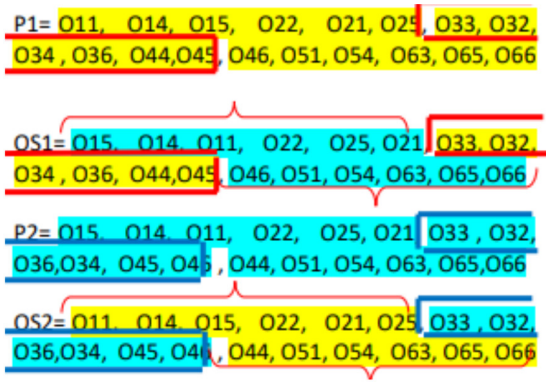

Fig. 14. Crossover of the fourth iteration

Mutation:
→ Mutation by exchange at the OS1 (Figure 15):
O44 ← O45
P1 = O15, O14, O11, O22, O25, O21, O33, O32, O34, O36, O45, O44, O46, O51, O54, O63, O65, O66

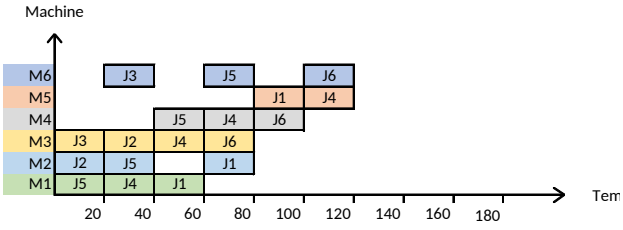

Fig. 15. Gantt chart after mutation at the OS1

T = 120 seconds
→ Mutation by exchange at the OS2 (Figure 16):
O11 ← O15
O21 ← O25
O46 ← O44

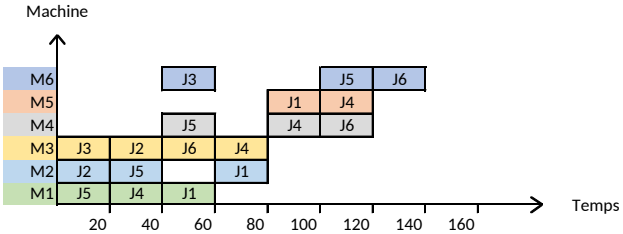

Fig. 16. Gantt chart mutation at the OS2

T = 140 seconds
P2 = O15, O14, O11, O22, O25, O21, O33, O32, O36, O34, O45, O44, O46, O51, O54, O63, O65, O66
The new population (fifth iteration):
P1 = O15, O14, O11, O22, O25, O21, O33, O32, O34, O36, O45, O44, O46, O51, O54, O63, O65, O66
P2 = O15, O14, O11, O22, O25, O21, O33, O32, O36, O34, O45, O44, O46, O51, O54, O63, O65, O66
The cross between the two parents P1 and P2 is shown in Figure 17.

Crossover:

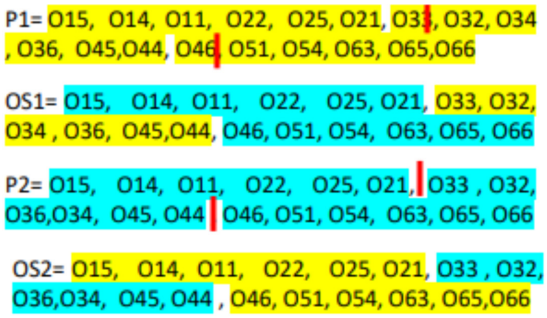

Fig. 17. Crossover of the fifth iteration

Mutation:
→ Mutation by exchange at the OS1 (Figure 18):
O15 ← O14
O33 ← O36
O32 ← O34
O45 ← O46
O65 ← O66
P1 = O14, O15, O11, O22, O25, O21, O36, O34, O32, O33, O46, O44, O45, O51, O54, O63, O66, O65

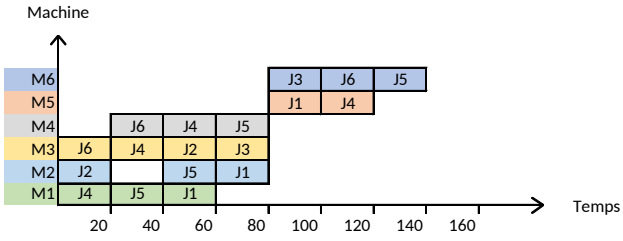

Fig. 18. Gantt chart after mutation

T = 140 seconds
→ Mutation by exchange at the OS2 (Figure 19):
O44 ← O46
P2 = O15, O14, O11, O22, O25, O21, O33, O32, O36, O34, O45, O46, O44, O51, O54, O63, O65, O66

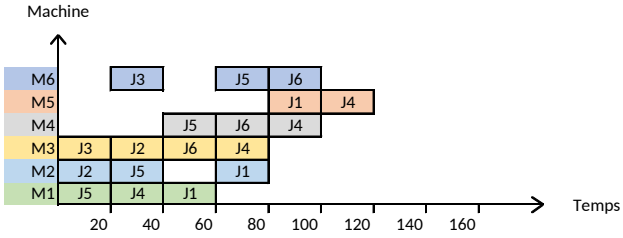

Fig. 19. Gantt chart after mutation

T=120 seconds
The new population (6th iteration):
P1 = O14, O15, O11, O22, O25, O21, O36, O34, O32, O33, O46, O44, O45, O51, O54, O63, O66, O65
P2 = O15, O14, O11, O22, O25, O21, O33, O32, O36, O34, O45, O46, O44, O51, O54, O63, O65, O66

Crossover:

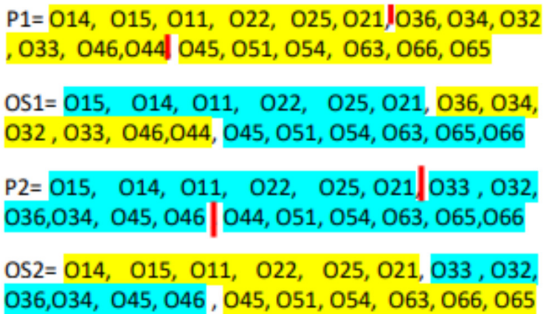

Fig. 20. Crossover of the sixth iteration

The cross between the two parents P1 and P2 is shown in Figure 20.
Mutation:
At the OS1 (Figure 21):
O15 ← O14
O32 ← O33
O51 ← O54
O65 ← O66
P1 = O14, O15, O11, O22, O25, O21, O36, O34, O33, O32, O46, O44, O45, O54, O51, O63, O66, O65

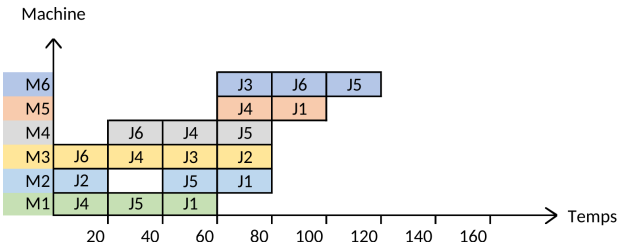

Fig. 21. Gantt chart after mutation

T = 120 seconds
At the OS2 = [O14, O15, O11, O22, O25, O21, O33, O32, O36,O34, O45, O46, O45, O51, O54, O63, O66, O65]
We choose to not proceed by a mutation, so we obtain:
P2 = OS2 = O14, O15, O11, O22, O25, O21, O33, O32, O36, O34, O45, O46, O45, O51, O54, O63, O66, O65

References

Asadzadeh, L., & Zamanifar, K., (2010). An agent-based parallel approach for the job shop scheduling problem with genetic algorithms. *Mathematical and Computer Modelling*, 52.

Báez, S., Angel-Bello, F., Alvarez, A., & Melián-Batista, B. (2019). A hybrid metaheuristic algorithm for a parallel machine scheduling problem with dependent setup times. *Computers & Industrial Engineering*, 131, 295–305.

Cheng, R., Gen, M., & Tsujimura, Y. (1996). A tutorial survey of job-shop scheduling problems. *Computers & Industrial Engineering*, 30(4), 983–997.

Chien, Ch-F., & Lan, Y-B. (2021). Agent-based approach integrating deep reinforcement learning and hybrid genetic algorithm for dynamic scheduling for Industry 3.5 smart production. *Computers & Industrial Engineering*, 162, 107782.

Cohan, F.M. (1984). Can Uniform Selection Retard Random Genetic Divergence Between Isolated Conspecific Populations? *Society for the Study of Evolution*, 38(3), 495–504.

Ghasemi, A., Ashoori, A., Heavey, C. (2021). Evolutionary Learning Based Simulation Optimization for Stochastic Job Shop Scheduling Problems. *Applied Soft Computing*, 106, 107309

Hoitomt, D.J., Luh, P.B., & Pattipati, K.R. (1993). A practical approach to job-shop scheduling problems. *IEEE Transactions on Robotics and Automation*, 9(1).

Kundakcı, N., & Kulak, O. (2016). Hybrid genetic algorithms for minimizing makespan in dynamic job shop scheduling problem. *Computers & Industrial Engineering*, 96.

Loukil, T., Teghem, J., Tuyttens, D. (2005). Solving multi-objective production scheduling problems using metaheuristics. *European Journal of Operational Research,* 161, 42–61.

Ritwik, K., & Deb, S. (2011). A genetic algorithm-based approach for optimization of scheduling in job shop environment. *Journal of Advanced Manufacturing Systems*, 10(2), 223–240. DOI: 10.1142/S0219686711002235.

Shen, K., De Pessemier, T., Martens, L., & Wout J. (2021). A parallel genetic algorithm for multi-objective flexible flowshop scheduling in pasta manufacturing. *Computers & Industrial Engineering*, 161, 107659.

Sun, K., Zheng, D., Song, H., Cheng, Z., Lang, X., Yuan, W., & Wang, J. (2023). Hybrid genetic algorithm with variable neighborhood search for flexible job shop scheduling problem in a machining system. *Expert Systems with Applications*, 215.

Swan, J., Adriaensen, S., Brownlee, A.E.I., Hammond, K., Johnson, C.G., Kheiri, A., Krawiec, F., Merelo, J.J., Minku, L.L., Özcan, E., Pappa, G.L., García-Sánchez, P., Sörensen, K., Voß, S., Wagner, M., & White, D.R. (2022). Metaheuristics "In the Large". *European Journal of Operational Research*, 297, 393–406.

Tamssaouet, K., Dauzère-Pérès, S., Knopp, S., Bitar, A., Yugma, C. (2022). Multiobjective optimization for complex flexible job-shop scheduling problems. *European Journal of Operational Research*, 296(1), 87–100.

Wein, L.M., Chevalier, P.B., (1992). *A Broader View of the Job-Shop Scheduling Problem.* Published Online: 1 Jul 1992. DOI: 10.1287/mnsc.38.7.1018.

Vilcot, G., & Billaut, J.-C. (2008). A tabu search and a genetic algorithm for solving a bicriteria general job shop scheduling problem. *European Journal of Operational Research*, 190.

Yu, H., & Liang, W. (2001). Neural network and genetic algorithm-based hybrid approach to expanded job-shop scheduling. *Computers & Industrial Engineering*, 39.