

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
MOHAMED BOUDIAF UNIVERSITY OF M'SILA



FACULTY OF MATHEMATICS AND INFORMATICS

DEPARTMENT OF INFORMATICS

End of study memory

Submitted for MASTER's degree

Domain: Mathematics and Informatics

Sector: Information technology

Speciality: Decision Informatics and Optimization

**By: BENNAOUI BOUCHRA
MANNED MAROUA**

Topic

**Machine Learning approach for single machine
scheduling problems**

The jury consists of:

Dr. HERAGUEMI KamelEddine	MBU of M'sila	President
Dr. THERAFI Abdellah	MBU of M'sila	Examiner
Dr. HEMMAK Allaoua	MBU of M'sila	Reporter

Academic year : 2021 /2022

Dedication

To our dear parents

Acknowledgment

Above all, thank you very much to God, we would like to thank our parents and siblings who have supported us during this academic journey and our entire lives.

We wish friends for their valuable assistance, continuous encouragement and continued moral support.

This work was never possible without the moral support and support of our families. We thank to express our gratitude and respect for Mr HammakAllaoua guidance.

We thank jury members for the honour they have given us by agreeing to evaluate and judge this humble act.

We thank our them all.

TABLE OF CONTENTS

GENERAL INTRODUCTION.....	11
CHAPTER I MACHINE LEARNING.....	12
1. INTRODUCTION.....	13
2. SCHEDULING	13
3. SINGLE MACHINE SCHEDULING.....	13
4. MACHINE LEARNING.....	13
5. THE TYPES OF MACHINE LEARNING	15
5.1 Supervised Learning.....	15
5.1.1 Classification.....	15
5.1.2 Regression.....	16
5.1.3 Forecasting.....	16
5.2. Semi-supervised learning.....	16
5.3. Unsupervised Learning	16
5.3.1 Clustering	17
5.3.2 Dimension reduction	17
5.4. Reinforcement learning.....	17
6. Machine Learning Modelling.....	18
6.1.Data Splitting.....	18
6.2.ML Model Implementation.....	18
6.3.Random Forest.....	18
6.4.XG Boost.....	18
6.5. ML Model Evaluation.....	19
7. Uses of machine learning algorithms.....	19
8. the most common and popular machine learning algorithms.....	19
9. Machine Learning-based scheduling.....	21
10. Experimental Study.....	22
11.Conclusion.....	23
CHAPTER 2SCHEDULING.....	24
1. INTRODUCTION	25
2. GENERAL INFORMATION OF SCHEDULING.....	25
3. ELEMENTS OF SCHEDULING PROBLEMS.....	25
3.1 The Task.....	25

3.2 The resource.....	26
3.2.1Renewable resources.....	26
3.2.2 Non-renewable resources.....	27
3.3 The Constraint.....	27
3.4 The criteria.....	27
4. SCHEDULING PROBLEMS CLASSIFICATIONS DIAGRAM	28
α : machine environment.....	28
β : the characteristics of the tasks.....	28
γ : the criterion(s) to be optimised.....	28
5. TYPES OF WORKSHOP SCHEDULING.....	28
5.1 Single machine workshop.....	29
5.2 Parallel machine workshop.....	29
5.3 Flow shop workshop	30
5.4Job shop workshop	30
5.5Open shop workshop	30
6. CONCEPT OF PROBLEM COMPLEXITY.....	31
6.1 The class NP.....	31
6.2 The class P.....	31
6.3 The NP-Hard class	31
6.4 The NP-Complete class.....	31
CONCLUSION	31
CHAPTER 3 STATE OF THE ART	33
1. INTRODUCTION	37
CHAPTER 4 MACHINE LEARNING FOR SCHEDULING PROBLEM ON SINGLE MACHINE.....	38
1. INTRODUCTION	39
2. PROBLEMATICAL	39
3. MATHEMATICAL FORMULATION	39
4. USED APPROACH.....	40
4.1 The algorithms.....	40
4.2 Dynamics programing.....	42
4.3 Machine learning.....	45
CHAPTER 5 : RESULT AND SUMMARY.....	52
1. INTRODUCTION	53
2. PHYSICAL ENVIRONMENT	53

3. SOFTWARE ENVIRONMENT	53
3.1 The programming language used.....	53
3.2 NetBeans IDE station.....	54
4. THE GENERAL INTERFACE OF OUR PROGRAM	55
5.TEST PHASE.....	55
5.1 First case_.....	55
5.2 Second case	59
5.3 Third case.....	62
6. MACHINE LEARNING STEPS	65
6.1 Collecting Data.....	65
6.2 Preparing the Data.....	66
6.3 Choosing a Model.....	66
6.4 Training the Model.....	66
6.5 Evaluating the Model.....	67
6.6 Parameter Tuning.....	67
6.7 Making Predictions	67
7. ANALYSIS OF RESULTS	67
8. CONCLUSION	68
GENERAL CONCLUSION	69
BIBLIOGRAPHY.....	70

LIST OF FIGURES

Figure 1.1. Machine learning general scheme

Figure 1.2.Supervised Learning

Figure 1.3. Semi-supervised learning

Figure 1.4. Unsupervised Learning

Figure 1.5. Reinforcement learning

Figure 1.6. Machine Learning-based Scheduling

Figure 2.1. Breakdown of resources.

Figure 2.2. Single Machine Scheduling.

Figure 2.3. Parallel machine scheduling.

Figure 2.4.Workshop flow shop scheduling.

Figure 2.5.Workshop schedule job shop.

Figure 5.1. The programming language java.

Figure 5.2.The NetBeans IDE software.

Figure 5.3 .The general interface of our program.

Figure 5.4. Results of dynamic programming, number jobs, due date ratio $n=3$, $r=0.2$.

Figure 5.5. Graphic curve for dynamic programming, number jobs, due date ratio, $n=3$, $r=0.2$. .

Figure 5.6. Results of dynamic programming, number jobs, due date ratio, $n=5$, $r=0.2$.

Figure 5.7. Graphic curve for dynamic programming, number jobs, due date ratio, $n=5$, $r=0.2$.

Figure 5.8 . Results of dynamic programming, number jobs, due date ratio, $n=100$, $r=0.2$.

Figure 5.9. Graphic curve for dynamic programming, number jobs, due date ratio, $n=100$, $r=0.2$.

Figure 5.10. Results of dynamic programming, number jobs, due date ratio, $n=50$, $r=0.4$.

Figure 5.11. Graphic curve for dynamic programming, number jobs, due date ratio, $n=50$, $r=0.4$.

Figure 5.12. Results of dynamic programming, number jobs, due date ratio, $n=1000$, $r=0.4$.

Figure 5.13 . Graphic curve for dynamic programming, number jobs, due date ratio ratio, $n=1000$, $r=0.4$.

Figure 5.14. Results of dynamic programming, number jobs, due date ratio $n=10$, $r=0.8$.

Figure 5.15 . Graphic curve for dynamic programming, number jobs, due date ratio, $n=10$, $r=0.8$.

Figure 5.14. Results of dynamic programming, number jobs, due date ratio, $n=10$, $r=0.8$.

Figure 5.15 . Graphic curve for dynamic programming, number jobs, due date ratio, $n=10$, $r=0.8$.

Figure 5.16. Results of dynamic programming, number jobs, due date ratio, $n=1000$, $r=0.8$.

Figure 5.17 .Graphic curve for dynamic programming, number jobs, due date ratio, $n=1000$, $r=0.8$.

GENERAL INTRODUCTION

Machine Learning is a rapidly growing research area for studying methods for developing AI systems that are capable of learning (Michalski). The ability to learn and improve is essential for intelligent systems; however, little work has been done in applying machine learning to intelligent scheduling (Shaw et al). Ayton et al. display a review of Machine Learning in scheduling.

From a computational view point, the problem of learning is to create computable procedures to perform tasks of which only partial descriptions about the way the system is desired to work are known. Scheduling is used to program the execution of tasks (operations) over a set of resources (human and machinery) in order to determine an optimal implementation plan, and to optimize certain time criteria, budget...etc.

Now, we are looking at how to optimize Makespan; there are many optimization methods that we choose machine-learning algorithms that they are derived from nature and they are most used in multiple domains. Machine learning algorithms are able to find solutions that may not be specific solutions.

We will also try to use the dynamic programming of machine learning algorithms to study the differences and extent of the effect of dynamic programming in accelerating the search for solutions.

Our brief will be divided into 5 chapters:

In the first chapter, we provide a definition and meaning of machine learning that also touches on its four types and how to use its algorithm.

In the second chapter we present in a general way the scheduling, we present the scheduling problem generally, then we explain the formulation of this problem, then we will represent in the fourth and fifth section classification and representation of scheduling problem. Finally, in the last section we will close this chapter by explaining the different methods of solving the scheduling problem.

In the third chapter is the state of the art we present what others do in our theme and present their work by a small abstract.

In the fourth chapter, we present how to solve the problem of dynamic programming and machine learning with the use of single machine algorithms.

Finally, in the last chapter we present our application and explain how we do it.

CHAPTER I

MACHINE LEARNING

1. Introduction

In this chapter, we provide a definition and meaning of machine learning that also touches on its four types and how to use its algorithm.

2. Scheduling

Most traditional approaches to solving scheduling problems use either simulation analysis, analytical models or combinations of these methods. Simulation appears to be the appropriate tool to analyse the effect of different releasing and dispatching policies (see, for example, Baker [2] and Stacked et al. [6]). The difficulty of the analytical approaches is the complexity of the scheduling problem as even the most simplified models for scheduling are NP hard problems which results in the impossibility of optimally solving realistically sized problems.

Over the last fifteen years, an important effort has been dedicated to developing Artificial Intelligence (AI) methods for scheduling. AI methods are often called knowledge-based systems. Machine learning lays in the field of AI.

3. Single machine scheduling

One of the most basic and fundamental scheduling problems is that of minimizing the number of tardy jobs on a single machine.

The following single machine scheduling problem is studied. There is a set

$N = \{1, \dots, n\}$ of jobs to be scheduled for processing on a single machine. Each job $j \in N$ has a processing time $p_j > 0$, a release time $r_j \geq 0$; and may have a due date $d_j \geq 0$ a weight $w_j > 0$ and a non-decreasing function $f_j(t)$ associated with it. The release times the earliest time when the job processing can start, the due date is the desired time for its completion, the weight indicates a relative importance of the job and the function $f_j(t)$ represents the cost that has to be paid when the job completes at time t . All data are assumed to be positive integer numbers and each cost function f_j , $j = 1, \dots, n$ is assumed to be computable in a constant time at every single point of its feasible domain.

4. Machine learning

Machine Learning is a rapidly growing research area for studying methods for developing AI systems that are capable of learning (Michalski [3]). The ability to learn and improve inessential

for intelligent systems; however, little work has been done in applying machine learning to intelligent scheduling (Shaw et al. [5]). Ayton et al. [1] display a review of Machine Learning in scheduling.

From a computational view point, the problem of learning is to create computable procedures to perform tasks of which only partial descriptions about the way the system is desired to work are known. In very general terms the machine learning scheme is depicted in figure 1.1

A particular case of machine learning is when the poorly structured information source of which the machine learning algorithms make use is given by a collection of examples which can be considered as a sample of the performance of that which is wanted to be learned.

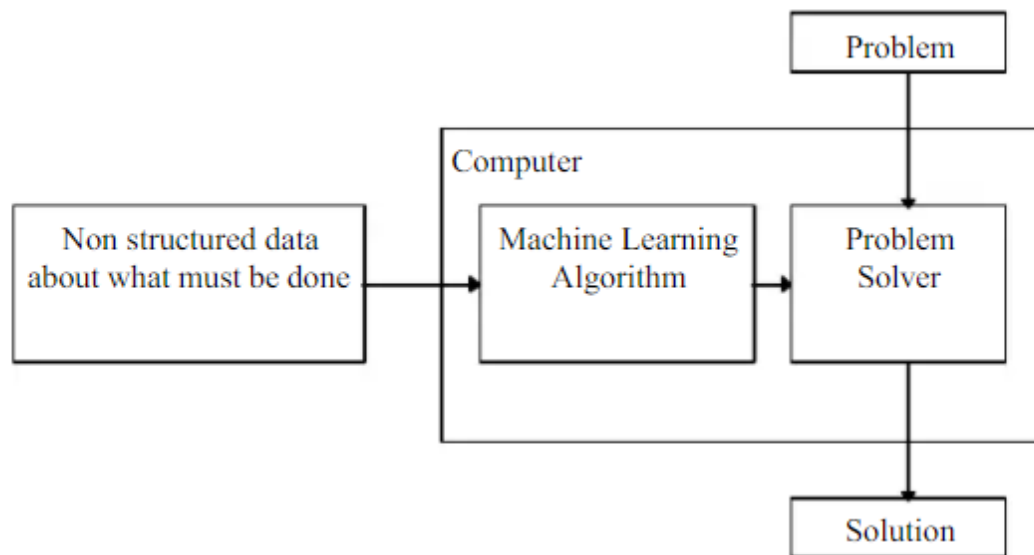


Figure 1.1. Machine learning general scheme.

Particularly, this examples, which are usually called training examples, will be described by attribute-value tables, where there will be a special attribute known as the class. The aim is to tyro learn to classify new cases similar to those in the examples, but about which only the values of the different attributes are known but not that of the class. Machine learning can be considered as a problem of creation in which knowledge is built. There is a great variety of knowledge representations, but the most used are the rules like the following:

IF $(b_{i1} \geq a_1 \geq c_{i1})$ AND..... $(b_{in} \geq a_n \geq c_{in})$ THAN c_i

Where a_j represents the j -th attribute, b_{ij} and c_{ij} define the range for a_j and C_j denotes the class and the trees of decision which main feature is the fact that the branches are used to represent questions about the values of the attributes and include in the leaves the decisions about the classes to which presumably the object to be classified will belong.

5. Types of machine learning approaches

5.1. Supervised Learning

In supervised learning, the machine is taught by example. The operator provides the machine learning algorithm with a known dataset that includes desired inputs and outputs, and the algorithm must find a method to determine how to arrive at those inputs and outputs. While the operator knows the correct answers to the problem, the algorithm identifies patterns in data, learns from observations and makes predictions. The algorithm makes predictions and is corrected by the operator – and this process continues until the algorithm achieves a high level of accuracy/performance.

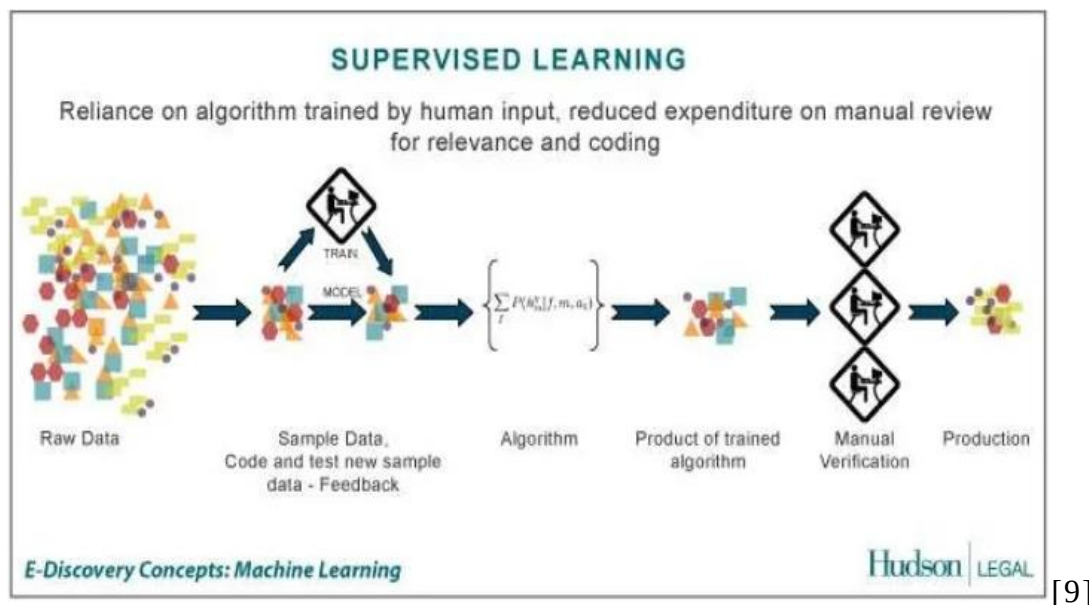


Figure 1.2. Supervised Learning.

Under the umbrella of supervised learning fall: Classification, Regression and Forecasting. [7]

5.1.1 Classification:

In classification tasks, the machine learning program must draw a conclusion from observed values and determine to what category new observations belong. For example, when filtering emails as ‘spam’ or ‘not spam’, the program must look at existing observational data and filter the emails accordingly.

5.1.2 Regression:

In regression tasks, the machine learning program must estimate – and understand – the relationships among variables. Regression analysis focuses on one dependent variable and a series of other changing variables – making it particularly useful for prediction and forecasting.

5.1.3 Forecasting:

Forecasting is the process of making predictions about the future based on the past and present data, and is commonly used to analyse trends.

5.2 Semi-supervised learning

Semi-supervised learning is similar to supervised learning, but instead uses both labelled and unlabelled data. Labelled data is essentially information that has meaningful tags so that the algorithm can understand the data, whilst unlabelled data lacks that information. By using this combination, machine learning algorithms can learn to label unlabelled data. [7]

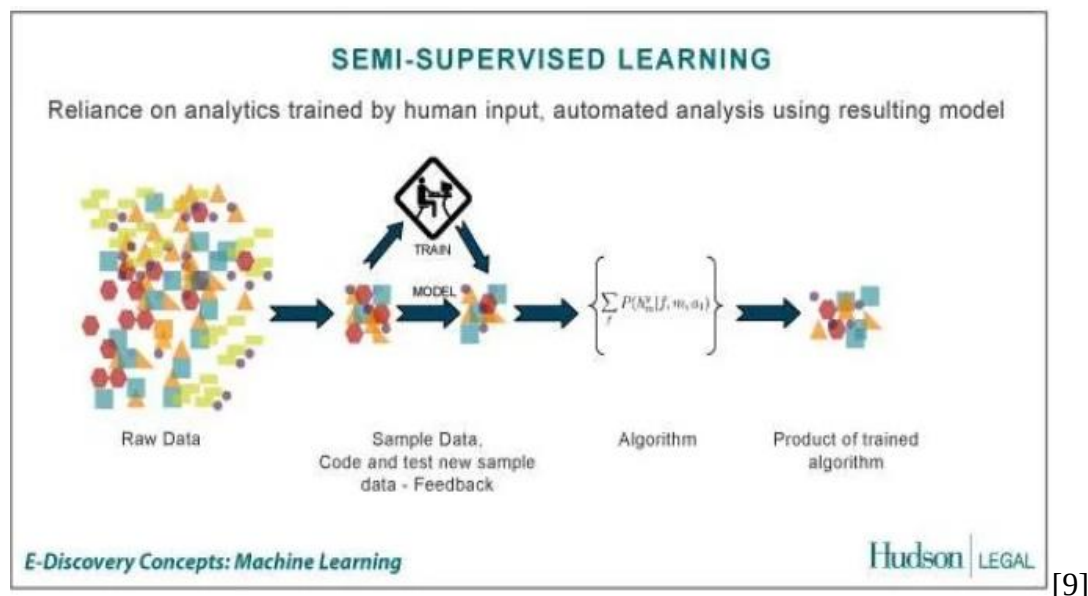


Figure 1.3. Semi-supervised learning

5.3 Unsupervised Learning

Here, the machine-learning algorithm studies data to identify patterns. There is no answer key or human operator to provide instruction. Instead, the machine determines the correlations and relationships by analysing available data. In an unsupervised learning process, the machine learning algorithm is left to interpret large data sets and address that data accordingly. The algorithm tries to organise that data in some way to describe its structure. This might mean grouping the data into clusters or arranging it in a way that looks more organised.

As it assesses more data, its ability to make decisions on that data gradually improves and becomes more refined.

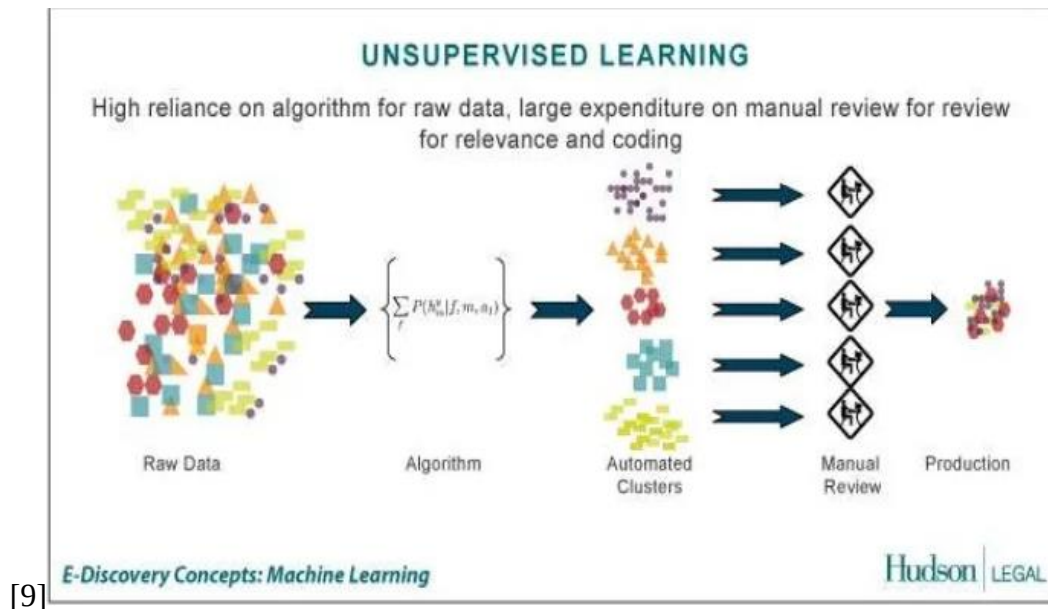


Figure 1.4. Unsupervised Learning

Under the umbrella of unsupervised learning, fall:

5.3.1 Clustering:

Clustering involves grouping sets of similar data (based on defined criteria). It's useful for segmenting data into several groups and performing analysis on each data set to find patterns.

5.3.2 Dimension reduction:

Dimension reduction reduces the number of variables being considered to find the exact information required.

5.4 Reinforcement learning

Reinforcement learning focuses on regimented learning processes, where a machine-learning algorithm is provided with a set of actions, parameters and end values. By defining the rules, the machine-learning algorithm then tries to explore different options and possibilities, monitoring and

evaluating each result to determine which one is optimal. Reinforcement learning teaches the machine trial and error. It learns from past experiences and begins to adapt its approach in response to the situation to achieve the best possible result [7].

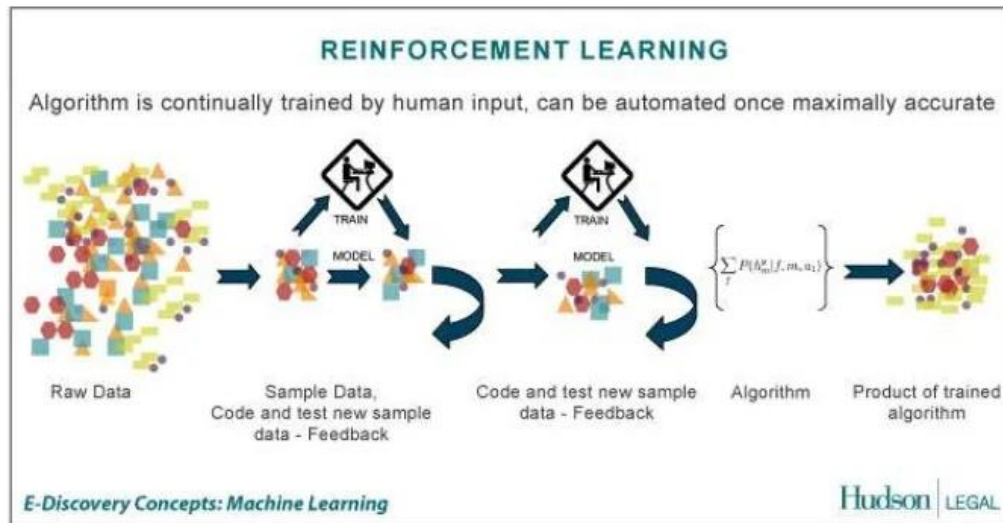


Figure 1.5. Reinforcement learning

6. Machine Learning Modelling

6.1 Data Splitting

The data is split 70% train data and 30% using the Train-Test-Split function. 70% of the data was for training used so as to maximize the prediction accuracy of the models.

6.2 ML Model Implementation

Two types of models were implemented for this machine learning regression task, the Random Forest model, and the Extreme Gradient Boosting model (XG Boost)

6.3 Random Forest

This machine-learning model is of the class of models known as bagging algorithms. These model improves performance accuracy by combining the results of several strong decision tree learners (that is they have a high low biases but high variance) in parallel such that at the end a much better decision tree learner with higher accuracy is obtained [8].

6.4 XG Boost

This belongs to the class of models known as boosting algorithms. This model improves the final decision tree learner by combining the capabilities of several weak learner trees (that is models with

higher bias but lower variance) with each other in a sequential manner such that at the end a much more robust decision tree learner is obtained.

6.5 ML Model Evaluation

Since the models are to predict a numeric value (trip duration in minutes) the Mean Absolute Error (MAE) was chosen as the evaluation metric of choice and the model that resulted in the lower MAE score was chosen as the preferred model to be used alongside the genetic algorithm [8].

7. Uses of machine learning algorithms

Choosing the right machine-learning algorithm depends on several factors, including, but not limited to: data size, quality and diversity, as well as what answers businesses want to derive from that data. Additional considerations include accuracy, training time, parameters, data points and much more. Therefore, choosing the right algorithm is both a combination of business need, specification, experimentation and time available. Even the most experienced data scientists cannot tell you which algorithm will perform the best before experimenting with others. We have, however, compiled a [machine learning algorithm ‘cheat sheet’](#), which will help you find the most appropriate one for your specific challenges.

8. The most common and popular machine learning algorithms

Naïve Bayes Classifier Algorithm (Supervised Learning - Classification)

The Naïve Bayes classifier is based on Bayes’ theorem and classifies every value as independent of any other value. It allows us to predict a class/category, based on a given set of features, using, and probability. Despite its simplicity, the classifier does surprisingly well and is often used due to the fact it outperforms more sophisticated classification methods.

K Means Clustering Algorithm (Unsupervised Learning - Clustering): The K Means Clustering algorithm is a type of unsupervised learning, which is used to categorise unlabelled data, i.e. data without defined categories or groups. The algorithm works by finding groups within the data, with the number of groups represented by the variable K. It then works iteratively to assign each data point to one of K groups based on the features provided.

Support Vector Machine Algorithm (Supervised Learning - Classification): Support Vector Machine algorithms are supervised learning models that analyse data used for classification and regression analysis. They essentially filter data into categories, which is achieved by providing a

set of training examples, each set marked as belonging to one or the other of the two categories. The algorithm then works to build a model that assigns new values to one category or the other.

Linear Regression (Supervised Learning/Regression) Linear regression is the most basic type of regression. Simple linear regression allows us to understand the relationships between two continuous variables.

Logistic Regression (Supervised learning – Classification) Logistic regression focuses on estimating the probability of an event occurring based on the previous data provided. It is used to cover a binary dependent variable, that is where only two values, 0 and 1, represent outcomes.

Artificial Neural Networks (Reinforcement Learning) An artificial neural network (ANN) comprises ‘units’ arranged in a series of layers, each of which connects to layers on either side. ANNs are inspired by biological systems, such as the brain, and how they process information. ANNs are essentially a large number of interconnected processing elements, working in unison to solve specific problems.

ANNs also learn by example and through experience, and they are extremely useful for modelling non-linear relationships in high-dimensional data or where the relationship amongst the input variables is difficult to understand.

Decision Trees (Supervised Learning – Classification/Regression) A decision tree is a flow-chart-like tree structure that uses a branching method to illustrate every possible outcome of a decision. Each node within the tree represents a test on a specific variable – and each branch is the outcome of that test.

Random Forests (Supervised Learning – Classification/Regression) Random forests or ‘random decision forests’ is an ensemble learning method, combining multiple algorithms to generate better results for classification, regression and other tasks. Each individual classifier is weak, but when combined with others, can produce excellent results. The algorithm starts with a ‘decision tree’ (a tree-like graph or model of decisions) and an input is entered at the top. It then travels down the tree, with data being segmented into smaller and smaller sets, based on specific variables.

Nearest Neighbours (Supervised Learning) The K-Nearest-Neighbour algorithm estimates how likely a data point is to be a member of one group or another. It essentially looks at the data points around a single data point to determine what group it is actually in. For example, if one point is on a grid and the algorithm is trying to determine what group that data point is in (Group A or Group B, for example) it would look at the data points near it to see what group the majority

of the points are in.

Clearly, there are a lot of things to consider when it comes to choosing the right machine learning algorithms for your business' analytics. However, you don't need to be a data scientist or expert statistician to use these models for your business. At SAS, our products and solutions utilise a comprehensive selection of machine learning algorithms, helping you to develop a process that can continuously deliver value from your data.

9. Machine Learning-based Scheduling

The suggested approach consists of five basic steps, as it is shown in figure1.2. The steps are the following: [5]

Definition of control attributes for capturing the relevant manufacturing patterns. Obviously it is not possible to take into account all possible attributes. A synthesis must be done.

Creation of training examples using different values of the control attributes. The devalues must be the common ones in the manufacturing system studied. For every training example, the rule of dispatching with the best performance, must be worked out through simulation. This rule will be the class for this training example.

Achieving heuristic rules by means of a learning program. One heuristic rule should look like the next one:

IF $(b_{i1} \geq a_1 \geq c_{i1})$ AND..... $(b_{in} \geq a_n \geq c_{in})$ THAN
rule dispatching

Using the previously calculated heuristic rules to select the most adequate dispatching rule depending on the set of values of the control attributes present in every moment.

To compare the performance of the heuristic rules versus the single dispatching rule which best performs in every moment. Should the first rules perform worse than the single dispatching rule, more training examples must be generated.[5]

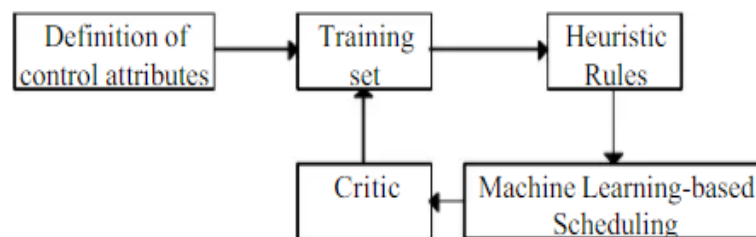


Figure 1.6. Machine Learning-based Scheduling**10.Experimental Study**

The system considered in this paper, is a simplified two work-centre flow shop:

WC1 and WC2. Each work centre has two machines capable of doing the same operations.

The set of dispatching rules used is the following:

SPT. The operations are ordered according to the shortest operation time first.

LPT. The operations are scored according to the longest operation time first.

EDD. The operations are ordered according to the earliest due date first.

FCFS. The operations are scored according to the rule first come first served The control attributes selected for generating the training examples are the following:

Mean arrival rate of jobs

Expected processing times on WC1

Expected processing times on WC2

Flow allowance factor (F) which measures due date tightness. Following Baker [2], the Total Work Content (TWK) rule for assigning job due dates has been employed. Under TWK, the due date d_i of job i is worked out as follows:

$d_i = a_i + F \times p_i$ Where a_i is the arrival time, and p_i is the processing time of the job i

The system performance criteria used in this paper is the mean tardiness, being these criteria the most used in all manufacturing systems. In the simulation experiments during the learning stage, it was supposed that the jobs arrived according to a Poisson distribution. The processing times followed an Exponential distribution and the Flow Allowance Factor varied within 2 to 4. Job interarrival times were varied to result in an overall system utilization within 60% to 95%. [5]

The experiments addressed 400 different combinations of these parameters. In each case, the steady state mean tardiness values resulting from employing the dispatching rules singularly were determined and the best one was chosen as the class for that combination. To obtain the heuristic rules it was used the learning program C4.5 (Quinlan [4]). The learning process in C4.5 follows a sequence of specialization steps guided by an information entropy function for evaluating class membership (Shaw et al. [5]). The number of heuristic rules obtained was 7. The EDD rule is chosen in 4 out of the 7 and the SPT one in the 3 left. The default class is SPT. [5]

Afterwards, various test simulations were generated by using new combinations of the parameters used in the training process and different number of changes in this combinations for each simulation. It was observed that in general terms the heuristic rules performed better than the best dispatching rules in each case. The number of test simulations were 30. The heuristic rules were better in 14 out of these 30, in 11 were as good and in 5 they were worse. The mean performance improvement was 8.5%. [5]

11. Conclusion

Popular machine learning applications and technology are evolving at a rapid pace, and we are excited about the possibilities that our Machine Learning course has to offer in the days to come. As the demand for AI and machine learning has increased, organizations require professionals with in-and-out knowledge of these growing technologies and hands-on experience. Keeping the innate need in mind, Simplilearn has launched AI and Machine Learning certification courses with Purdue University in collaboration with IBM that will help you gain expertise in various industry skills and technologies from Python, NLP, speech recognition, to advanced deep learning. This Post Graduate program will help you stand out in the crowd and grow your career in thriving fields like AI, machine learning, and deep learning.

CHAPTER II

SCHEDULING

1. Introduction

In the industrial field, scheduling involves finding an optimal solution by schedule the execution of tasks by simultaneously allocating the necessary resources to perform these tasks over time and within certain production constraints (delay, priorities, linking constraints).

In this chapter, we present the basic concepts of production, the concepts related to scheduling, namely the definition of a problem planning, the objectives of the scheduling and the types of workshops as well as the concept of the complexity of the optimization problem.

2. General information of scheduling

Scheduling the operation of an industrial production system consists of managing the allocation of resources over time, while at the same time optimising a set of criteria. It is also programming the execution of an achievement by allocating resources to tasks and setting dates for completion. [10]

Scheduling may also involve scheduling the execution of operations in allocating the required resources and setting their manufacturing start dates.

The sequencing process consists of three steps:

- Planning: which aims to determine the various operations to be carried out, the dates and the material and human means to be used.
- The execution, which consists in implementing the various operations defined in the planning phase.
- Control, which consists in making a comparison between planning and execution, either at the cost level or at the implementation date level.

3. Elements of scheduling problems

Scheduling problems appear in all areas: IT, industry, construction, administration, etc. [11].

The different data of a scheduling problem are tasks, resources, and constraints and criteria. Thus, given a set of tasks and a set of resources, it is a question of schedule tasks and allocate resources to optimize one or more objectives (a target corresponding to a performance criterion) respecting a set of constraints.

3.1 The Task

A task is an elementary entity located in time, by a start date and/or end, and the realisation of which requires a predefined duration. It consists of a series of operations which require certain resources and that it is necessary to program in order to optimize a certain objective [11].

There are two types of tasks:

- split (pre-emptible) tasks that can be performed in multiple runs, thus facilitating the resolution of certain problems,
- Non-matching (indivisible) tasks that must be completed in one go and are not interrupted until completed.

3.2 The Resource

A resource is a technical or human means, intended to be used for the accomplishment of a task and available in limited quantities. Generally, $M = \{M_1, M_2, \dots, M_m\}$ all resources. Several types of resources need to be distinguished [10].

A resource type decomposition is illustrated in Figure 1.

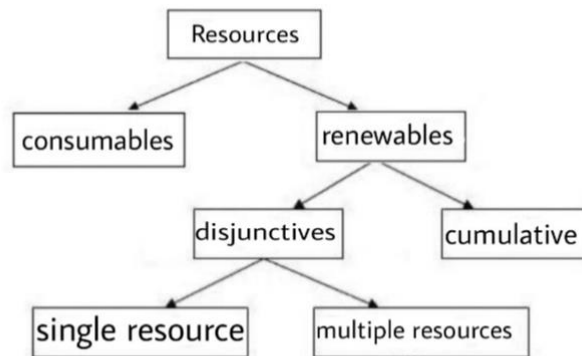


Figure 2.1. Breakdown of ressources.

3.2.1 Renewable resources

A renewable resource is a resource that is used by a task and remains available after the end of the task in the same quantity for use by the remaining tasks, for example renewable resources: machines, employees, etc.

- Disjunctive Resources

Are non-shareable resources, they run by one task at a time?

- Cumulative Resources

Are sharable resources, they execute by multiple tasks simultaneously.

3.2.2 Non-renewable resources

A resource is non-renewable (or consumable), is a resource consumed completely over time by a certain spot so will not be available for use by the other remaining spots, for example: the raw material is a consumable resource.

3.3 The Constraint

Depending on the availability of resources and the evolution over time, two types of constraints can be distinguished [11]: resource constraints and time constraints.

- Resource constraints: several types of constraints can be induced by the nature of the resources. For example, the limited capacity of a resource not to exceed a certain number of tasks to be performed on this resource.

Resource constraints can be disjointed, inducing a task performance constraint on disjointed time intervals for a same resource, or cumulative involving the limitation of the number of tasks to carry out in parallel.

- Time constraints: they represent restrictions on the values that may take some scheduling time variables. These constraints may be:
 - Deadline constraints, some tasks must be completed before a predetermined date,
 - of precedence constraints, a task i must precede task j , time constraints at the earliest, linked to the unavailability of certain factors necessary to start performing tasks.

3.4 The criteria

A criterion corresponds to qualitative and quantitative requirements to be met in order to evaluate the quality of the established scheduling.

The criteria depending on a given application are very numerous; several criteria can be selected for the same application. The choice of the most satisfactory solution depends on the criteria previously defined, which can be classified into two types, regular and irregular. The different criteria are not independent; some are even equivalent.

Two criteria are equivalent if an optimal solution for one is also optimal for the other and conversely [11]

- Regular criteria are decreasing functions of completion dates operations. Some examples are listed below:
 - Minimisation of the completion dates of the actions,

- Minimisation of the maximum completion dates of the actions,
- Minimisation of the average completion dates of the actions,
- Minimisation of delays in the completion dates of actions,
- Minimisation of the maximum delays in the completion dates of the actions.
- Irregular criteria are non-standard criteria, that is, not monotonous functions of the operations end dates, such as:
 - Minimisation of outstanding's,
 - minimising the cost of storing raw materials,
 - balancing of machine loads,
 - Optimisation of tool changes.

The satisfaction of all criteria at once is often delicate, as it often leads to conflicting situations and seeking solutions to problems optimization complexes.

4. Scheduling Problems classifications diagrams

We follow the classification schemes proposed by (Graham et al, 1979) [12].

Three-field classification α , β , γ :

- α : machine environment.
 - α : Refers to the number of machines. If α is positive integer, the number of machines assumed constant. If α is absent then this number is assumed to be arbitrary.
- β : the characteristics of the tasks.
 - β_1 = pmtn if pre-emption of tasks is allowed, otherwise β_1 is absent.
 - If there are precedence constraints between tasks β_2 = prec, otherwise β_2 is empty.
 - β_3 = rj if the start dates at the earliest rj (or availability dates) of the tasks do not are not necessarily identical, otherwise ($j, rj = 0$) β_3 is absent.
- γ : the criterion(s) to be optimised.

5. Types of workshop scheduling

The classification of scheduling problems in a workshop can be done according to the number of machines and their order of use to manufacture a product, which depends on the nature of the workshop in question [12]. A workshop is characterized by the number of machines it contains and by his type.

5.1 Single-machine workshop

Each job consists of a single operation and each operation can be performed by one machine [13]. Figure 2 represents the workshop.

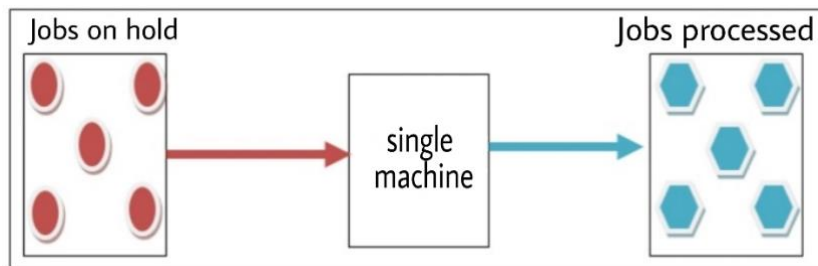


Figure 2.2. Single Machine Scheduling.

5.2 Parallel machine workshop

Each job consists of a single operation and each operation can be performed by any of the machines, arranged in parallel, but requires only one [13].

An example of the workshop is illustrated in Figure 3.

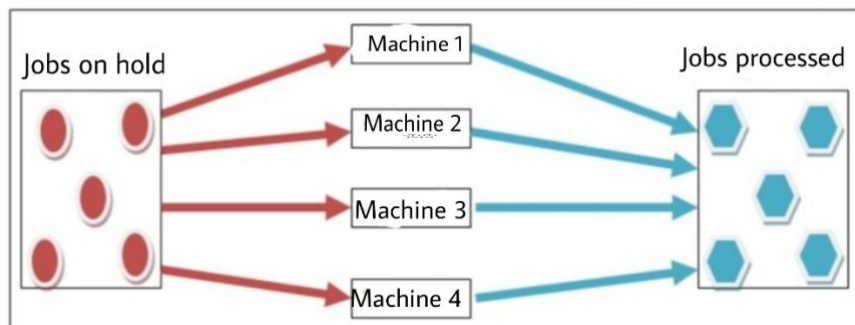


Figure 2.3. Parallel machine scheduling.

5.3 Flow-shop workshop

Also known as one-way workshops, they are considered as a line of which consists of several machines in series; all the operations of all the tasks go through the machines in the same order. In

flow-shop type workshops hybrid, a machine can exist in several identical copies running in parallel. Figure 4 is an example of the flow shop workshop[13].

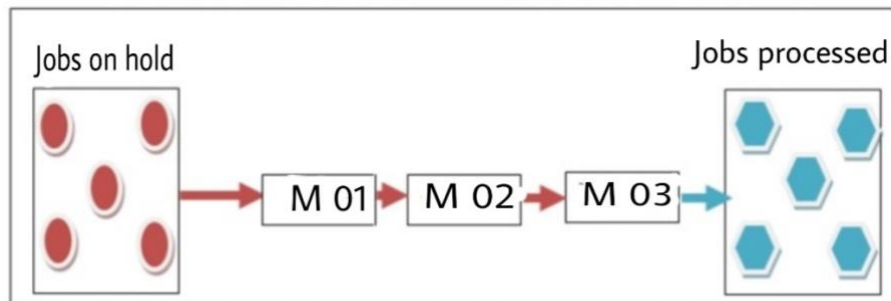


Figure 2.4. Workshop flow shop scheduling.

5.4 Job shop workshop

Also called multi-path workshops; on this type the order of passage of jobs on machines can be challenging from one job to another. There are possibilities of passing a job on a machine several times, so the chain of these jobs on the job shop workshops is a loop [13].

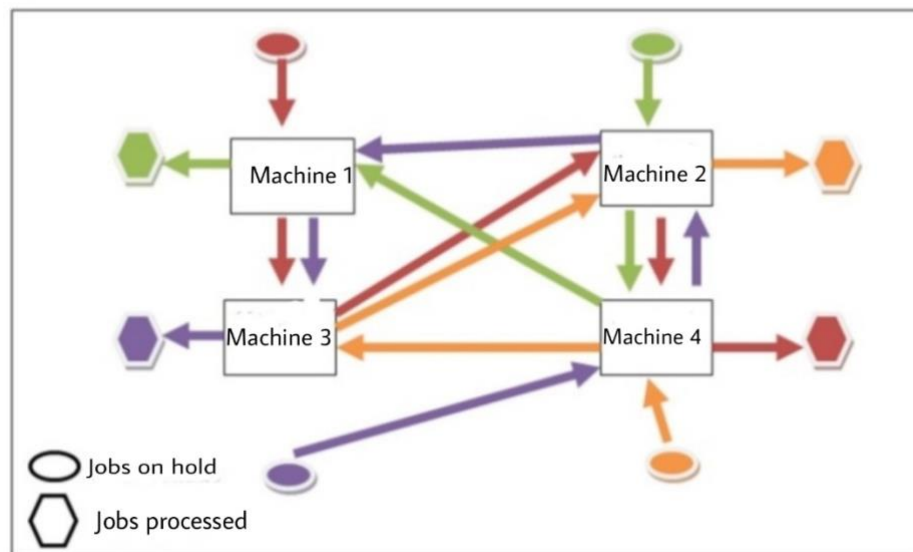


Figure 2.5.Workshop schedule job shop.

5.5 Open-shop workshop

These workshops are generally less constrained than those of the flow-shop type or job shop type. Thus, the order of operations is not fixed a priori; the scheduling problem on the one hand, to determine the path of each product and, on the other hand, to schedule the products taking into

account the ranges found, these two problems can be solved simultaneously. Compared to other workshop models, the open shop is not used easily in companies [13].

6. Concept of problem complexity

Concept of complexity of problems the theory of complexity aims to provide information on the difficulty a problem to be solved. It makes it possible to classify "mathematically" the problems according to their difficulty [10].

Solution. To solve a scheduling problem, the existence of a solution, or build it, because the difficulty lies around building the solution more that the improver of its existence and that it helps us to classify problems as being difficult or easy.

6.1 The class NP

In this class we find the decision problems for which there are nondeterministic polynomial algorithms they process [10]. This class contains two subclasses: the NP-Complete class and the NP-Difficult class

6.2 The class P

These are polynomial problems if there is a polynomial complexity algorithm for their resolution [10].

Class P is the set of all polynomial recognition problems. For the rest of the NP class, we are not sure that there is not a polynomial algorithm to solve each of its problems. Thus, we know that P is included in NP but we could not prove that P is not NP.

6.3 The NP-Complete class

A problem is NP-complete when it can be solved by a restricted class and that it can be used to simulate any other problem with a similar algorithm [10]. ...

The set of NP-complete problems is often designated by NP-C or NPC.

6.4 The NP-Hard class

In theoretical computing, a NP-difficult problem is a problem towards which we can reduce any problem of the NP class by a polynomial reduction [10].

If it is also in the NP class, it is said that it is a NP-complete problem

7. Conclusion

In this chapter we identified the production system, the scheduling problem defined by a set of tasks to be performed on a range of resources (machines) so that an objective function is improved within certain constraints.

Scheduling can be applied to any improved problem category easy or hard to get good results.

CHAPTER III

STATE OF THE ART

Introduction:

In this chapter, we represent articles by researchers doing research in our theme of the brief, then summarizes each article as follows:

(2019) Matthias Putz& Andreas Schlegel:

Use an online single machine environment is used to investigate the application potential of reinforcement learning for job shop scheduling problems, to quantify states and actions, to define reward functions, and to examine their influences on the performance of the algorithm through plenty of experiments. The findings from this research are expected to be the basis for further investigations into applying reinforcement learning to more complex job shop environments in the future.

(2010) JulienPerez:

In this theme, the use of artificial learning for the problem of sequencing is used in high performance calculation grids. After recalling and justifying the need for activity in these infrastructures, we have demonstrated the difficulties of batch scheduling systems commonly uses to provide satisfactory quality of service.

We have demonstrated that the use of explicit objective functions, as stated in the general approach of autonomic computing and more practically in the paradigm of learning by Reinforcement, effectively addresses this issue. From now on, we have taken the step of formalizing this issue of scheduling reactive as a multi-objective reinforcement learning issue in the field This formalization has led us to make a number of choices as to a description of the Markov decision-making processes to be adopted.

These are the difficult ones "to those modelisation choices that have led us to consider the use of a model" The European Commission's The European Commission's Directorate-General for on the environment and the action of the consultation process the use of a model. Experiments have shown that in a set of synthetic and synthetic cases, a model, the Echo State Network, offers better results than neural resells a retro gradient propagation today the state of the art in this field Finally, he proposed dividing the debate process into two problems: independents: the placement of jobs and the dynamic dimensioning of the infrastructure It is therefore possible for us to be able to the use of the calculation site. The use of eel traces from a calculation site of the EGEE grid allowed us to prove the contribution of this approach to performance currently offered by the most commonly used scheduler at the EGEE Grid, PBS.

Concerning the analysis of the problem of discrimination, theseThe results of the experiments have also enabled us to show that theEquity and then use were relevant gain functions in this context.

(2020) YuanyuanLi,EdoardoFadda, Daniele Manerba, Roberto Tadei, and Olivier Terzo:

Online scheduling has been an attractive field of research for over three decades. Some recent developments suggest that Reinforcement Learning (RL) techniques have the potential to deal with online scheduling issues effectively.

Driven by an industrial application, in this paper we apply four of the most important RL techniques, namely Q-learning, Sarsa, Watkins's $Q(\lambda)$, and Sarsa(λ), to the online single-machine scheduling problem.

Our main goal is to provide insights on how such techniques perform. The numerical results show that Watkins's $Q(\lambda)$ performs best in minimizing the total tardiness of the scheduling process.

(2021) Axel Parmentiera, Vincent T'Kindt:

they focus on the solution of a hard single machine scheduling problem by new heuristic algorithms embedding techniques from machine learning field and scheduling theory. These heuristics transform an instance of the hard problem into an instance of a simpler one solved to optimality.

The obtained schedule is then transposed to the original problem. Computational experiments show that they are competitive with state-of-the-art heuristics, notably on large instances.

(2007) Zhanguo Zhu, Ming Liu, Feng Chu,Yunfei Fang:

They study single-machine scheduling problem with time-dependent delivery times and learning effect. The delivery time of a job is proportional to its waiting time in the system, i.e., the length of the already scheduling jobs. That is, the delivery times are past-sequence-dependent (psd). Delivery time denotes the time a job required to be delivered to the customer.

It is assumed that the learning process reflects a decrease in the process time as a function of the number of repetitions, i.e., as a function of the job position in the sequence. they respectively consider the following objectives: the minimization of the make span, the total completion time, the total absolute differences in completion times and the sum of earliness, tardiness and common due date penalty. We develop polynomial time algorithms to optimally solve these problems.

(2019/2020)KACIMI EL HASSANI AICHA ABASSI HAYAT :

In the industrial environments as in the production, the very competing commercial context obliges the companies to diversify their offers for better answering the needs of their customers. Managing this diversity has become a central issue, as well as optimizing the supply chain became one of the

major issues of the companies in recent years. Our objective was to develop a tool to analyze the structure of the supply chain and its efficiency.

This tool is based on the combination of the two approaches optimization and simulation it contains an optimization module based on genetic algorithms and a simulation module allowing the evaluation of the studied chain during the optimization.

(2016) HANNACHE ABOUBAKER LEMMOUIS ABDELHAMID:

This work concerns solving a type of scheduling problem with Job Shop transmission constraint in which the products are transported in a unidirectional sense.

This problem is NP-hard problem is for this reason that there are no exact methods for solving large instances of this type of problem. Our goal in this work is to try to adapt one of Metaheuristics techniques for solving this problem all by minimizing the total time of execution Make span. An algorithm based on Genetic Algorithms method is proposed to give more close to the optimum solutions.

The proposed algorithm is then evaluated through a series of examples from the literature. The computer simulation in Matlab allows us to deduce the performance and efficiency of our algorithm especially for problems of small sizes.

(2020/2021) Sahel Facial:

The scheduling problems of a job shop type production system are often rated NP-Difficult. Solving such problems requires dedicated methods, while exact methods cannot solve these types of problems due to the huge computational time, heuristics offer the possibility of finding a workable solution in a reasonable time.

In this work we studied the performance of three heuristics used for solving this type of problem, namely SPT, LPT and shifting bottleneck heuristics. From the results we have observed that the heuristic shifting bottleneck provided satisfactory results for small size problems, and for medium and large size problems the LPT heuristic (with machine availability) gave the best results.

(1998) Dirk Biskup:

The focus of this work is to analyze learning in single-machine scheduling problems. It is surprising that the well-known learning effect has never been considered in connection with scheduling problems.

It is shown in this paper that even with the introduction of learning to job processing times two important types of single-machine problems remain polynomially solvable.

CHAPTER IV

MACHINE LEARNING FOR SCHEDULING PROBLEM

1. Introduction

In this chapter we expose the idea to solve a single machine scheduling problem with the use of machine learning and dynamic programming algorithms we try to maximize the Makespan.

2. Problematical

Given an array of jobs where every job has a deadline and associated profit if the job is finished before the deadline. It is also given that every job takes a single unit of time, so the minimum possible deadline for any job is one. How to maximize total profit if only one job can be scheduled at a time.

Examples:

Input: Four Jobs with following
Deadlines and profits

JobID	Deadline	Profit
-------	----------	--------

A	4	20
---	---	----

B	1	10
---	---	----

C	1	40
---	---	----

D	1	30
---	---	----

Output: Following is maximum
Profit sequence of jobs

C, A

Input: Five Jobs with following
Deadlines and profits

JobID	Deadline	Profit
-------	----------	--------

A	2	100
---	---	-----

B	1	19
---	---	----

C	2	27
---	---	----

D	1	25
---	---	----

E	3	15
---	---	----

Output: Following is maximum
Profit sequence of jobs

C, A, E

3. Mathematical Formulation

Following is the algorithm.

- 1) Sort all jobs in decreasing order of profit.
- 2) Iterate on jobs in decreasing order of profit. For each job, do the following:
 - a) Find a time slot i , such that slot is empty and $i < \text{deadline}$ and i is greatest. Put the job in this slot and mark this slot filled.
 - b) If no such i exists, then ignore the job.

4. Used Approach

4.1 The algorithms

- This program solves a problem of sequencing independent tasks on a single machine with a common expired date d . Given the number of tasks n , for each task i , profit $profit(i)$ and processing time $primer(i)$.
- The goal is to maximize the sum of the profits of the tasks without delay.
- The method used is dynamic programming.

```
public class SCHEDULING {  
  
    // A utility function that returns maximum of two integers  
    static int max(int a, int b)  
    { return (a > b) ? a : b; }  
  
    // Returns the maximum profit sum  
  
  
  
    // Driver program to test above function  
    public static void main(String args[])  
    {  
        int n = 5;
```

```
int profit[] = new int[] { 60, 100, 120, 50 , 250 };
int prtime[] = new int[] { 10, 20, 30, 41,65 };
int sumprtime = 0;
for(int i=0; i<n;i++) sumprtime+= prtime[i];
double ratio = 0.5;
int d = (int) ((int) sumprtime*ratio);

int i, w;
int K[][] = new int[n + 1][d + 1];

// Build table K[][] in bottom up manner
for (i = 0; i<= n; i++) {
    for (w = 0; w<= d; w++) {
        if (i == 0 || w == 0)
            K[i][w] = 0;
        else if (prtime[i - 1]<= w)
            K[i][w] = max(profit[i - 1] + K[i - 1][w - prtime[i - 1]], K[i - 1][d]);
        else
            K[i][w] = K[i - 1][w];
    }
}

System.out.println("The maximum profit sum is : " + K[n][d]);
System.out.println(" Jobs in time : " );

while (n != 0) {
    if (K[n][d] != K[n - 1][d]) {
        System.out.println("\tJob " + n + " with Processing time = " + prtime[n - 1] + " and
Profit = " + profit[n - 1]);
        d = d - prtime[n-1];
    }
}
```

```
        n--;  
    }  
    }  
}
```

4.2. Dynamic programing

Dynamic programming is a programming principle where a very complex problem can be solved by dividing it into smaller subproblems. This principle is very similar to recursion, but with a key difference, every distinct subproblem has to be solved only once.

To understand what this means, we first have to understand the problem of solving recurrence relations. Every single complex problem can be divided into very similar subproblems; this means we can construct a recurrence relation between them [14].

Examples:

4.2.1 Dynamic programming

Given a binary matrix of R rows and C columns.

We are allowed to flip to any size of sub matrix. Flipping means changing 1 to 0 and 0 to 1. The task is to maximize the number of 1s in the matrix. Output the maximum number of 1s.

Input: R = 3, C =3

```
mat[][] = { 0, 0, 1,  
            0, 0, 1,  
            1, 0, 1}
```

Output: 8

Flip

0 0 1

0 0 1

1 0 1

to get

1 1 1

1 1 1

0 1 1

Input: R = 2, C = 3

mat[][] = { 0, 0, 0,

0, 0, 0}

Output: 6

Create a matrix ones [][] of R rows and C columns, which precomputes the number of ones in the submatrix from (0, 0) to (i, j) by

// Common elements in ones [i-1][j] and

// ones[i][j-1] are ones[i-1][j-1]

ones[i][j] = ones[i-1][j] + ones[i][j-1] -

ones[i-1][j-1] + (mat[i][j] == 1)

Since, we are allowed to flip sub matrix only once. We iterate over all possible submatrix of all possible sizes for each cell (i, j) to (i + k - 1, i + k - 1). We calculate the total number of ones after the digits are flipped in the chosen submatrix.

Total number of ones in the final matrix after flipping submatrix (i, j) to (i + k - 1) will be Ones in the whole matrix - Ones in the chosen submatrix + Zeroes in the chosen sub matrix. That comes out to be-

ones[R][C] - cal(i, j, i + k, j + k - 1) + k*k - cal(i, j, i + k - 1, j + k - 1)

where cal(a, b, c, d) denotes the number of ones in square submatrix of length c - a.

Now cal(x1, y1, x2, y2) can be define by:

ones[x2][y2] - ones[x2][y1 - 1] - ones[x1 - 1][y2] + ones[x1 - 1][y1 - 1].

Below is the implementation of this approach:

```
// Java program to find maximum number of ones after
// one flipping in Binary Matrix
class GFG {

static final int R = 3;
static final int C = 3 ;

// Return number of ones in square submatrix of size
// k x k starting from (x, y)
static int cal(int ones[][], int x, int y, int k)
{
    return ones[x + k - 1][y + k - 1] - ones[x - 1][y + k - 1]
        - ones[x + k - 1][y - 1] + ones[x - 1][y - 1];
}

// Return maximum number of 1s after flipping a submatrix
static int sol(int mat[][])
{
    int ans = 0;
    int val = 0;
    // Precomputing the number of 1s
    int ones[][] = new int[R + 1][C + 1];
    for (int i = 1; i <= R; i++)
        for (int j = 1; j <= C; j++) {
            if(mat[i - 1][j - 1] == 1)
                val=1;
            ones[i][j] = ones[i - 1][j] + ones[i][j - 1] -
                ones[i - 1][j - 1] +
                (val);
        }
}
```

```
// Finding the maximum number of 1s after flipping
for (int k = 1; k <= Math.min(R, C); k++)
    for (int i = 1; i + k - 1 <= R; i++)
        for (int j = 1; j + k - 1 <= C; j++)
            ans = Math.max(ans, (ones[R][C] + k * k -
                                2 * cal(ones, i, j, k)));

return ans;
}

// Driver code
static public void main(String[] args) {
    int mat[][] = {{0, 0, 1},
                   { 0, 0, 1},
                   { 1, 0, 1 }
    };

    System.out.println(sol(mat));
}
}
```

// This code is contributed by Rajput-Ji

Output: 8

Time Complexity: $O(R \cdot C \cdot \min(R, C))$

.43. Machine learning

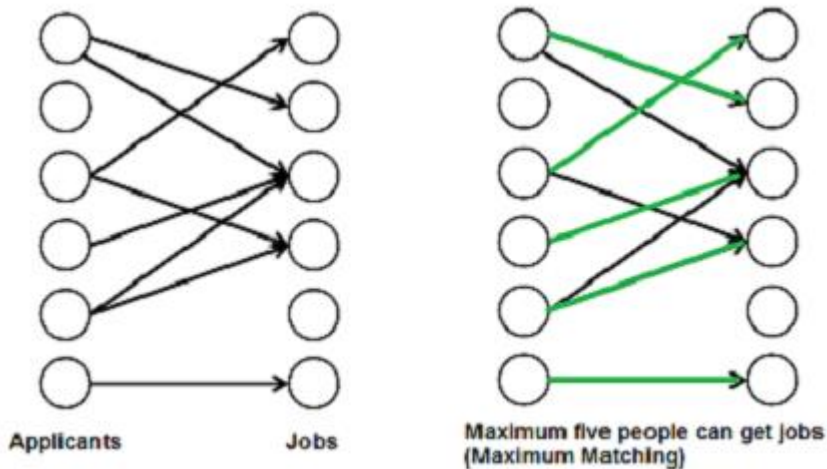
- **Maximum Bipartite Matching**

A matching in a Bipartite Graph is a set of the edges chosen in such a way that no two edges share an endpoint. A maximum matching is a matching of maximum size (maximum number of edges). In a maximum matching, if any edge is added to it, it is no longer a matching. There can be more than one maximum matchings for a given Bipartite Graph [14].

Why do we care?

There are many real world problems that can be formed as Bipartite Matching. For example, consider the following problem:

There are M job applicants and N jobs. Each applicant has a subset of jobs that he/she is interested in. Each job opening can only accept one applicant and a job applicant can be appointed for only one job. Find an assignment of jobs to applicants in such that as many applicants as possible get jobs.

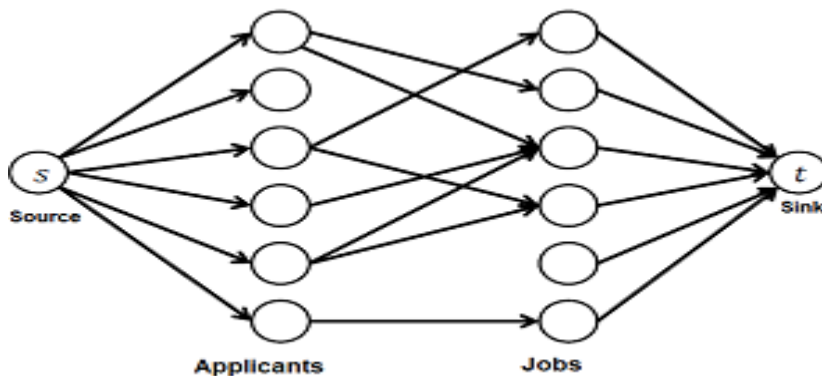


We strongly recommend to read the following post first.

[Ford-Fulkerson Algorithm for Maximum Flow Problem](#)

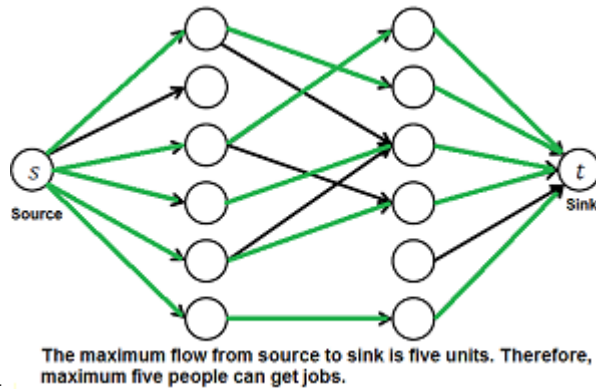
[Maximum Bipartite Matching and Max Flow Problem](#)

Maximum Bipartite Matching (MBP) problem can be solved by converting it into a flow network (See this video to know how did we arrive this conclusion). Following are the steps.



1) Build a Flow Network

There must be a source and sink in a flow network. So we add a source and add edges from source to all applicants. Similarly, add edges from all jobs to sink. The capacity of every edge is marked



as 1 unit.

2) Find the maximum flow.

We use Ford-Fulkerson algorithm to find the maximum flow in the flow network built in step 1. The maximum flow is actually the MBP we are looking for.

How to implement the above approach?

Let us first define input and output forms. Input is in the form of Edmonds matrix which is a 2D array 'bpGraph[M][N]' with M rows (for M job applicants) and N columns (for N jobs). The value bpGraph[i][j] is 1 if i'th applicant is interested in j'th job, otherwise 0.

Output is number maximum number of people that can get jobs.

A simple way to implement this is to create a matrix that represents adjacency matrix representation of a directed graph with $M+N+2$ vertices. Call the fordFulkerson() for the matrix. This implementation requires $O((M+N)*(M+N))$ extra space.

Extra space can be reduced and code can be simplified using the fact that the graph is bipartite and capacity of every edge is either 0 or 1. The idea is to use DFS traversal to find a job for an applicant (similar to augmenting path in Ford-Fulkerson). We call bpm() for every applicant, bpm() is the DFS based function that tries all possibilities to assign a job to the applicant.

In bpm(), we one by one try all jobs that an applicant 'u' is interested in until we find a job, or all jobs are tried without luck. For every job we try, we do following.

If a job is not assigned to anybody, we simply assign it to the applicant and return true. If a job is assigned to somebody else say x, then we recursively check whether x can be assigned some other job. To make sure that x doesn't get the same job again, we mark the job 'v' as seen before we

make recursive call for x. If x can get other job, we change the applicant for job 'v' and return true. We use an array maxR[0..N-1] that stores the applicants assigned to different jobs.

If bpm() returns true, then it means that there is an augmenting path in flow network and 1 unit of flow is added to the result in maxBPM().

// A Java program to find maximal

// Bipartite matching.

import java.util.*;

import java.lang.*;

import java.io.*;

class GFG

{

 // M is number of applicants

 // and N is number of jobs

 static final int M = 6;

 static final int N = 6;

 // A DFS based recursive function that

 // returns true if a matching for

 // vertex u is possible

 boolean bpm(boolean bpGraph[][], int u,
 boolean seen[], int matchR[])

 {

 // Try every job one by one

 for (int v = 0; v < N; v++)

 {

 // If applicant u is interested

 // in job v and v is not visited

 if (bpGraph[u][v] && !seen[v])

 {

```
// Mark v as visited
seen[v] = true;

// If job 'v' is not assigned to
// an applicant OR previously
// assigned applicant for job v (which
// is matchR[v]) has an alternate job available.
// Since v is marked as visited in the
// above line, matchR[v] in the following
// recursive call will not get job 'v' again
if (matchR[v] < 0 || bpm(bpGraph, matchR[v],
                        seen, matchR))
{
    matchR[v] = u;
    return true;
}
}
return false;
}
```

```
// Returns maximum number
// of matching from M to N
int maxBPM(boolean bpGraph[][])
{
    // An array to keep track of the
    // applicants assigned to jobs.
    // The value of matchR[i] is the
    // applicant number assigned to job i,
    // the value -1 indicates nobody is assigned.
    int matchR[] = new int[N];
```

```
// Initially all jobs are available
for(int i = 0; i < N; ++i)
    matchR[i] = -1;

// Count of jobs assigned to applicants
int result = 0;
for (int u = 0; u < M; u++)
{
    // Mark all jobs as not seen
    // for next applicant.
    boolean seen[] =new boolean[N] ;
    for(int i = 0; i < N; ++i)
        seen[i] = false;

    // Find if the applicant 'u' can get a job
    if (bpm(bpGraph, u, seen, matchR))
        result++;
}
return result;
}

// Driver Code
public static void main (String[] args)
    throws java.lang.Exception
{
    // Let us create a bpGraph shown
    // in the above example
    boolean bpGraph[][] = new boolean[][]{
        {false, true, true,
        false, false, false},
```

```
{true, false, false,
true, false, false},
{false, false, true,
false, false, false},
{false, false, true,
true, false, false},
{false, false, false,
false, false, false},
{false, false, false,
false, false, true}}};

GFG m = new GFG();
System.out.println( "Maximum number of applicants that can"+
    " get job is "+m.maxBPM(bpGraph));
}
}
```

Output:

Maximum number of applicants that can get job is 5.

CHAPTER V
RESULTS AND DISCUSSION

1. Introduction

As with many algorithms, the parameters of the machine learning and dynamic programming algorithm must be carefully chosen to achieve good results, so in this chapter we will perform tests to know which are the right parameters of the machine learning and dynamic programming algorithm must opt to provide good solutions or solutions close to the optimum of the single machine problem.

2. Physical Environment

We have used as a hardware environment an HP computer has the following characteristics:

- Processor: Intel® Core™ i5-6300U CPU @ 2.40GHz
- RAM: 8.00 Go
- System Type: 64-bit operating system.

3. Software Environment

Platform used is Windows 10 with the help of the following program:

3.1 The programming language used

We use this program Java is a general purpose, high-level programming language first released by Sun Microsystems in 1995. It is designed to have as few implementation dependencies as possible, is free to use, and can run on all platforms. It is concurrent, class-based, and object-oriented. Simply put, Java is a computing platform where users can develop applications.

Java is similar to C++ but is simplified to eliminate language features that cause common programming errors. The source code files, meaning files with a .java extension, are compiled into a format known as bytecode, which are files with a .class extension. This can then be executed by a Java interpreter. Bytecode can be directly converted into machine language instructions by a just-in-time compiler [15].

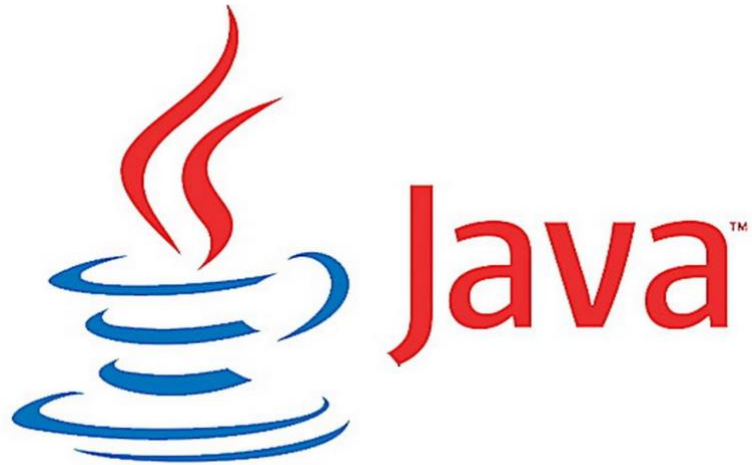


Figure 5.1 The programming language Java.

3.2 NetBeans IDE station

We use this program to create several computers have deferential features to test only our program. This program is the intoxication of devastation with language java.

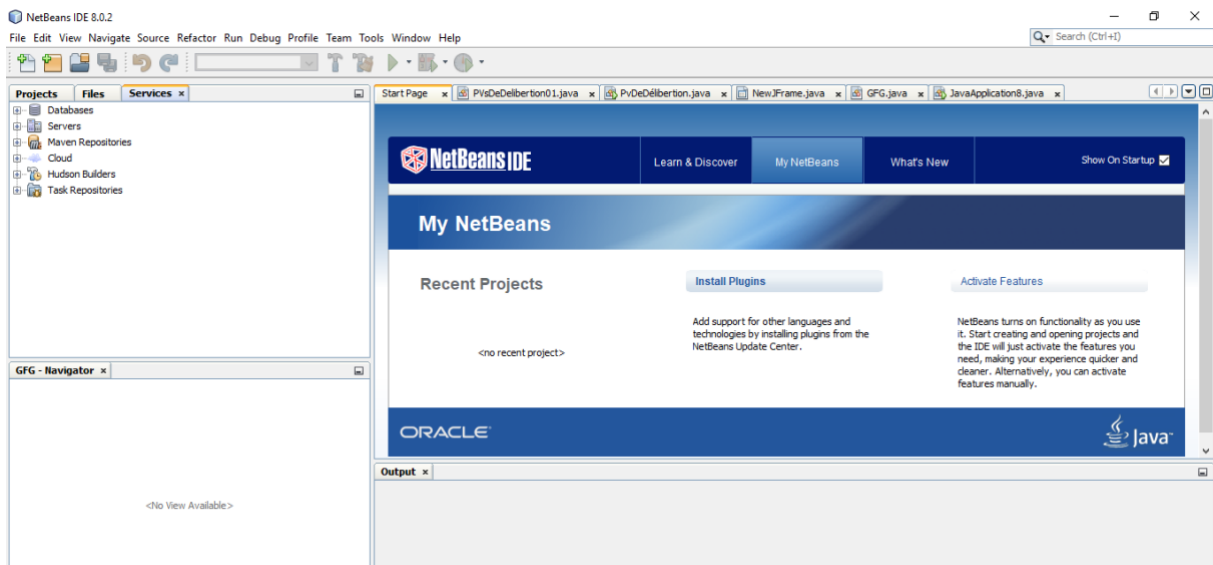


Figure 5.2 The NetBeans IDE software.

4. The general interface of our program

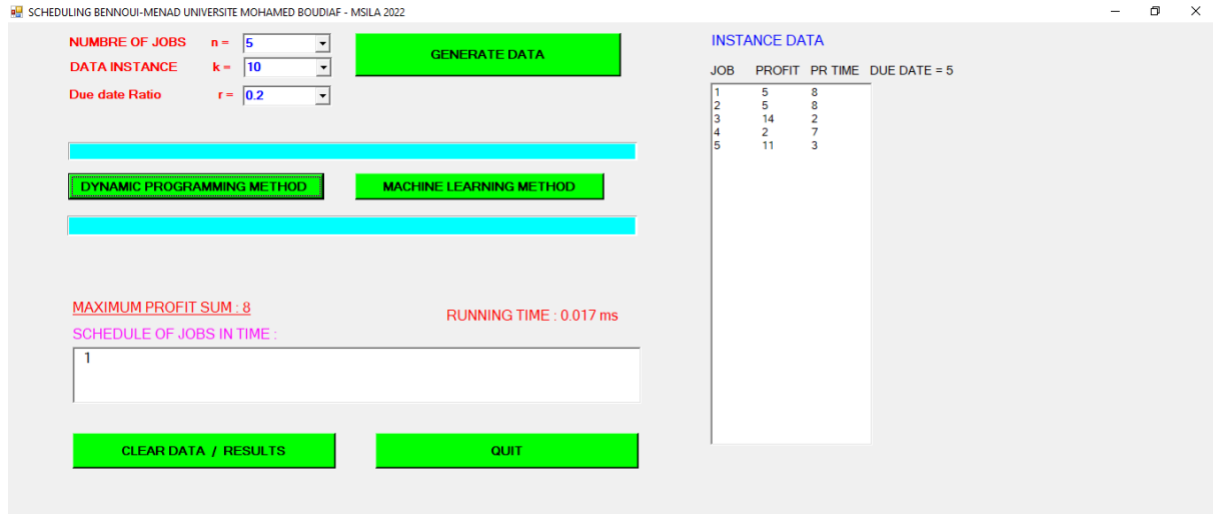


Figure 5.3 The general interface of our program.

5. Test phase

5.1 First case

dynamic programming:

- Capacity (ratio): $r = 0.2$.

Example 01:

Number of jobs: $n = 3$.

For the above parameters, the following results are obtained:

Instance	Maximum Profit Sum
1	0
2	9
3	0
4	0
5	1
6	0
7	7
8	10
9	0
10	0
Average	2.7

Tab 5.1. Dynamics programming.

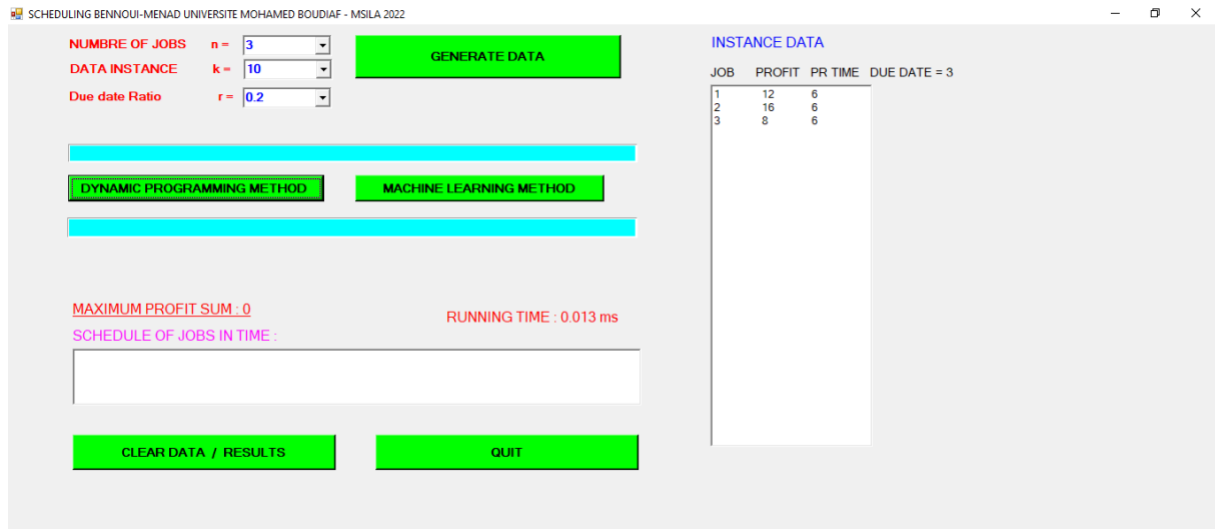


Figure 5.4:result of dynamic programming, number jobs, duo date ratio, $n=3$. $r=0.2$.

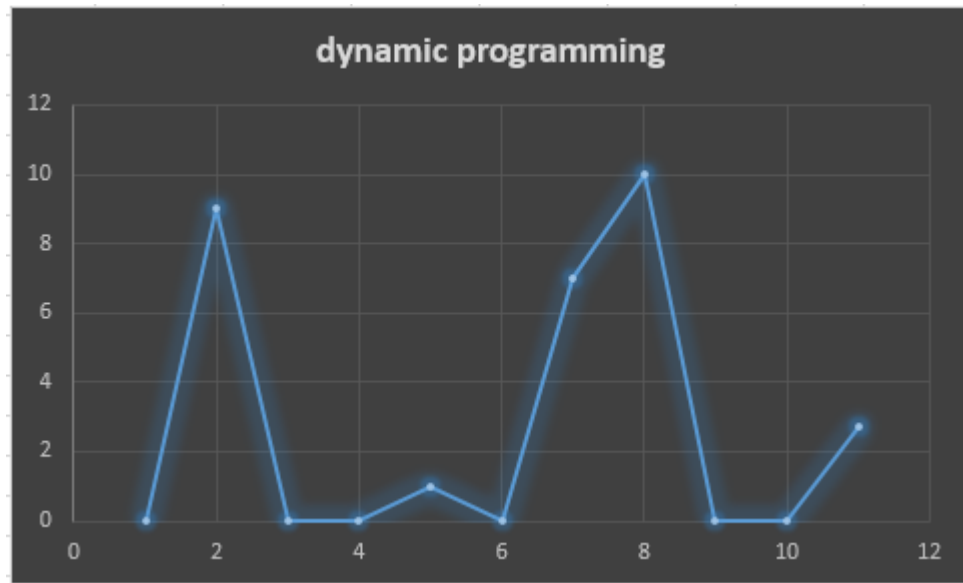


Figure 5.5:Graphic curve for dynamic programming, number jobs, duo date ratio, $n=3$. $r=0.2$.

Example 02:

- ✓ Number of jobs: $n=5$.

For the above parameters, the following results are obtained:

Instance	Maximum Profit Sum
1	8
2	10
3	7
4	0
5	7
6	10
7	13
8	10
9	4
10	8
Average	7.7

Tab 5.2. Dynamics programming.

SCHEDULING BENNOUI-MENAD UNIVERSITE MOHAMED BOUDIAF - MSILA 2022

NUMBRE OF JOBS n = 5
 DATA INSTANCE k = 10
 Due date Ratio r = 0.2

GENERATE DATA

DYNAMIC PROGRAMMING METHOD
 MACHINE LEARNING METHOD

MAXIMUM PROFIT SUM : 8
 SCHEDULE OF JOBS IN TIME : 1
 RUNNING TIME : 0.017 ms

CLEAR DATA / RESULTS QUIT

INSTANCE DATA

JOB	PROFIT	PR TIME	DUE DATE = 5
1	5	8	
2	5	8	
3	14	2	
4	2	7	
5	11	3	

Figure 5.6:result of dynamic programming, number jobs, duo date ratio, n=5. r=0.2.

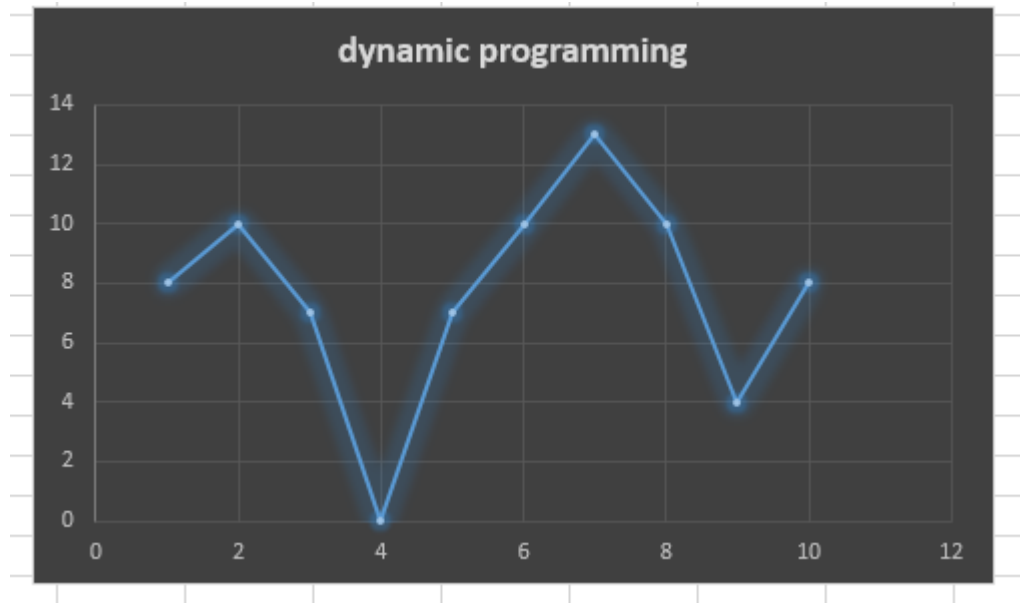


Figure 5.7:Graphic curve for dynamic programming, number jobs, duo date ratio, $n=5$. $r=0.2$.

Example 03:

- ✓ Number of jobs: $n = 100$.

For the above parameters, the following results are obtained:

Instance	Maximum Profit Sum
1	206
2	165
3	198
4	154
5	243
6	212
7	176
8	226
9	216
10	207
Average	200.3

Tab 5.3. Dynamics programming.

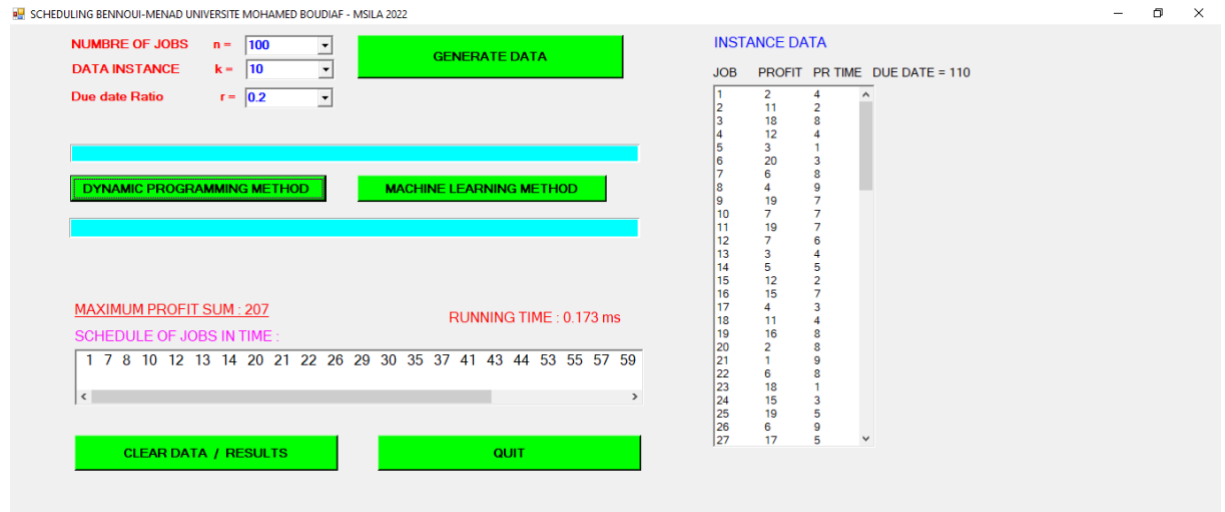


Figure 5.8:result of dynamic programming, number jobs, duo date ratio, $n=100$. $r=0.2$.

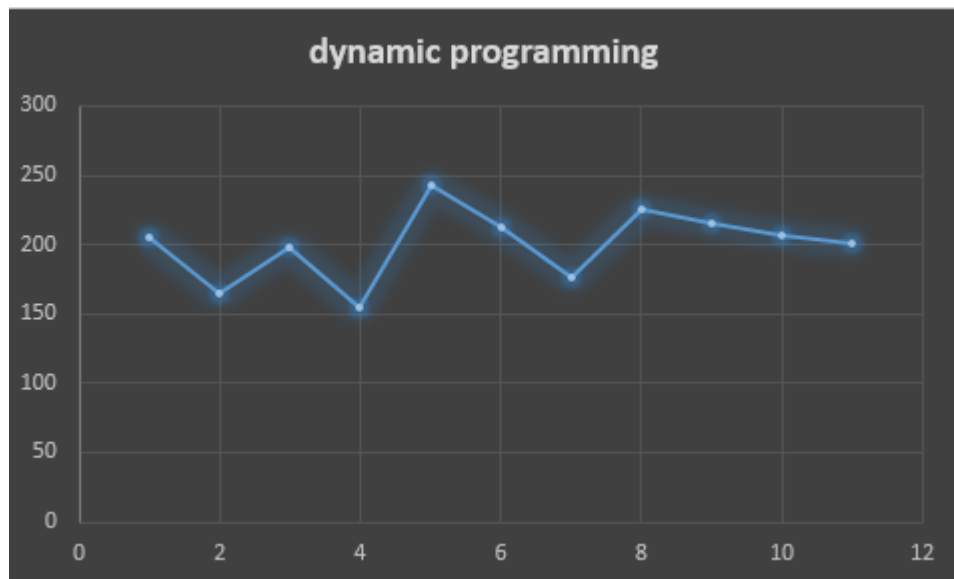


Figure 5.9:Graphic curve for dynamic programming, number jobs, duo date ratio, $n=100$. $r=0.2$.

5.2 Second case

Dynamic programming:

- Capacity (ratio): $r = 0.4$.

Example 01:

Number of jobs: $n = 50$.

For the above parameters, the following results are obtained:

Instance	Maximum Profit Sum
1	141
2	156
3	138
4	136
5	117
6	154
7	107
8	111
9	160
10	132
Average	135.2

Tab 5.4. Dynamics programming.

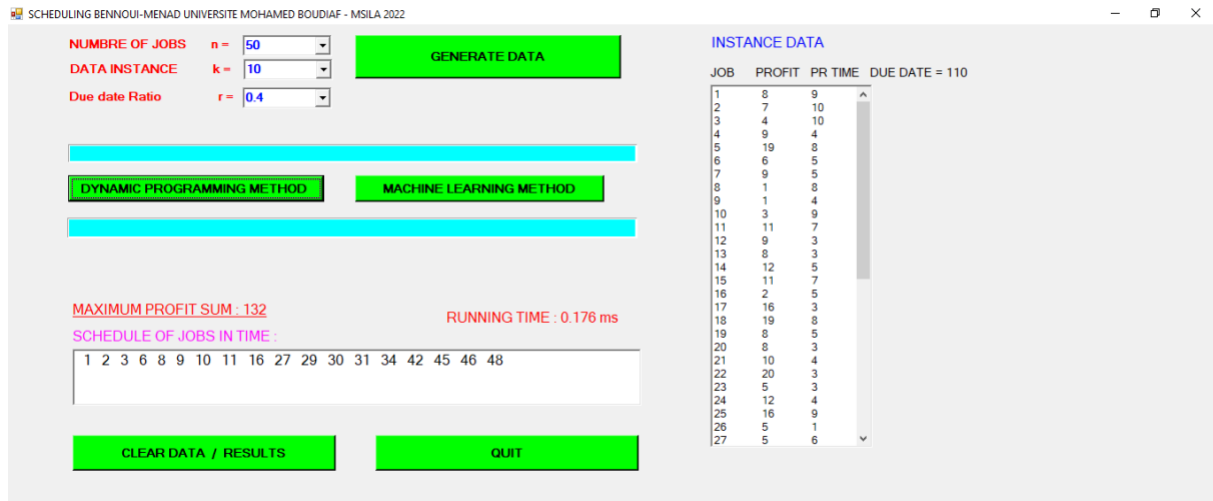


Figure 5.10:result of dynamic programming, number jobs, duo date ratio, n=50. r=0.4.

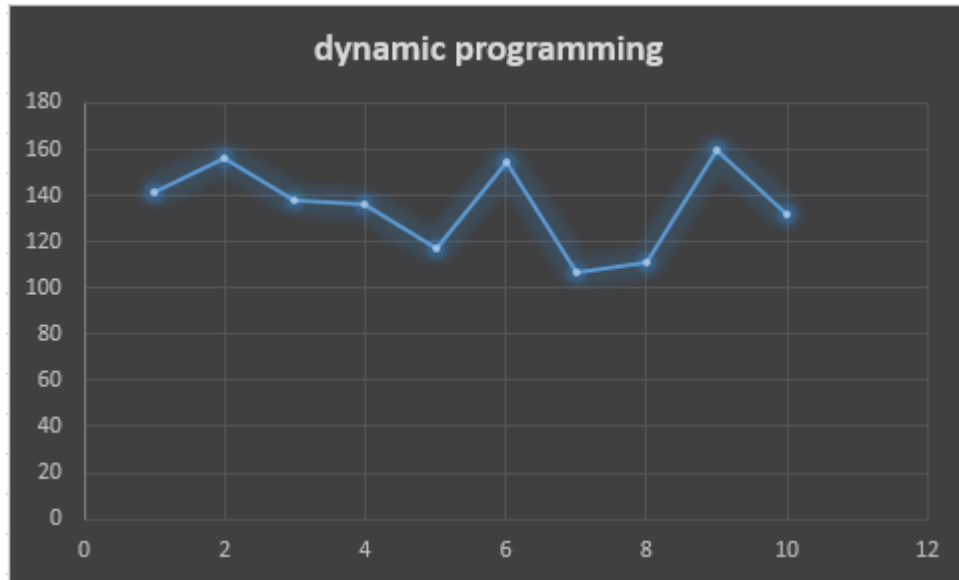


Figure 5.11: Graphic curve for dynamic programming, number jobs, duo date ratio, $n=50$. $r=0.4$.

Example 02:

Number of jobs: $n=1000$.

For the above parameters, the following results are obtained:

Instance	Maximum Profit Sum
1	2903
2	2719
3	2717
4	2706
5	2746
6	2747
7	2772
8	2843
9	2790
10	2882
Average	2782.5

Tab 5.5. Dynamics programming.

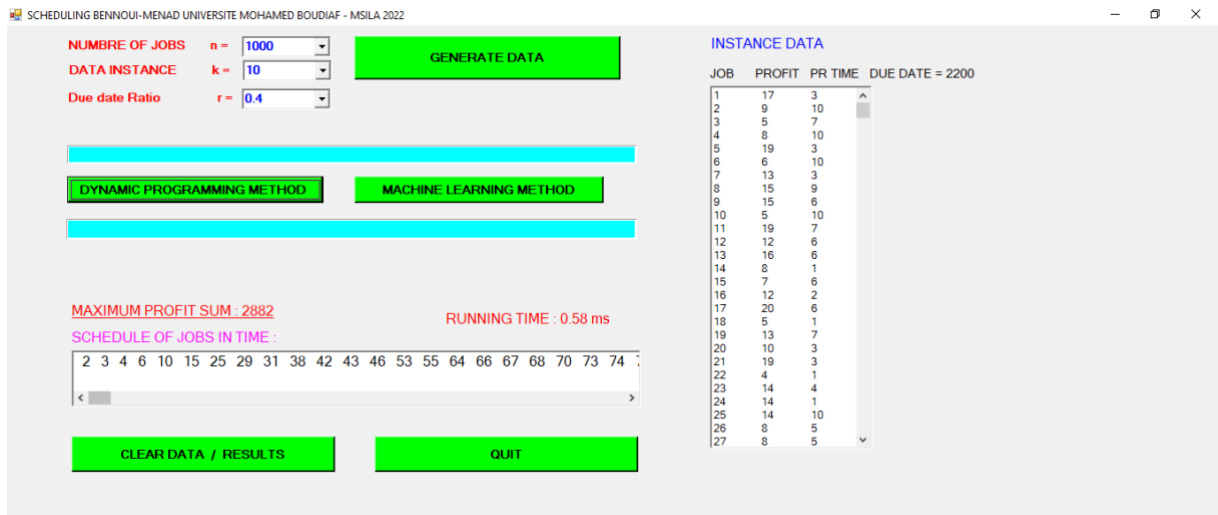


Figure 5.12: result of dynamic programming, number jobs, duo date ratio, $n=1000$. $r=0.4$.

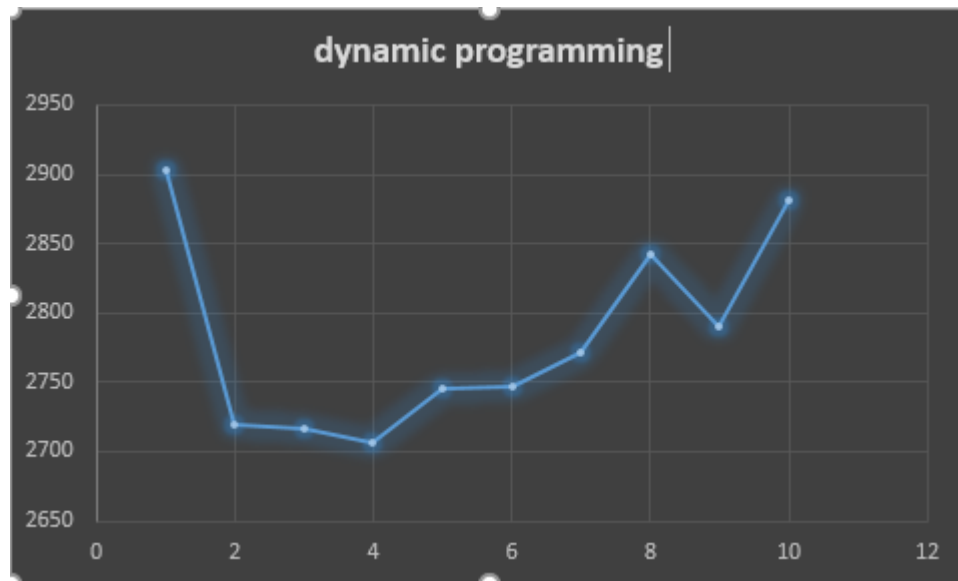


Figure 5.13: Graphic curve for dynamic programming, number jobs, duo date ratio, $n=1000$. $r=0.4$.

5.3 Third case

Dynamic programming:

- Capacity (ratio): $r = 0.8$.

Example 01:

Number of jobs: $n = 50$.

For the above parameters, the following results are obtained:

Instance	Maximum Profit Sum
1	40
2	45
3	41
4	34
5	32
6	51
7	29
8	57
9	43
10	33
Average	40.5

Tab 5.6. Dynamics programming.

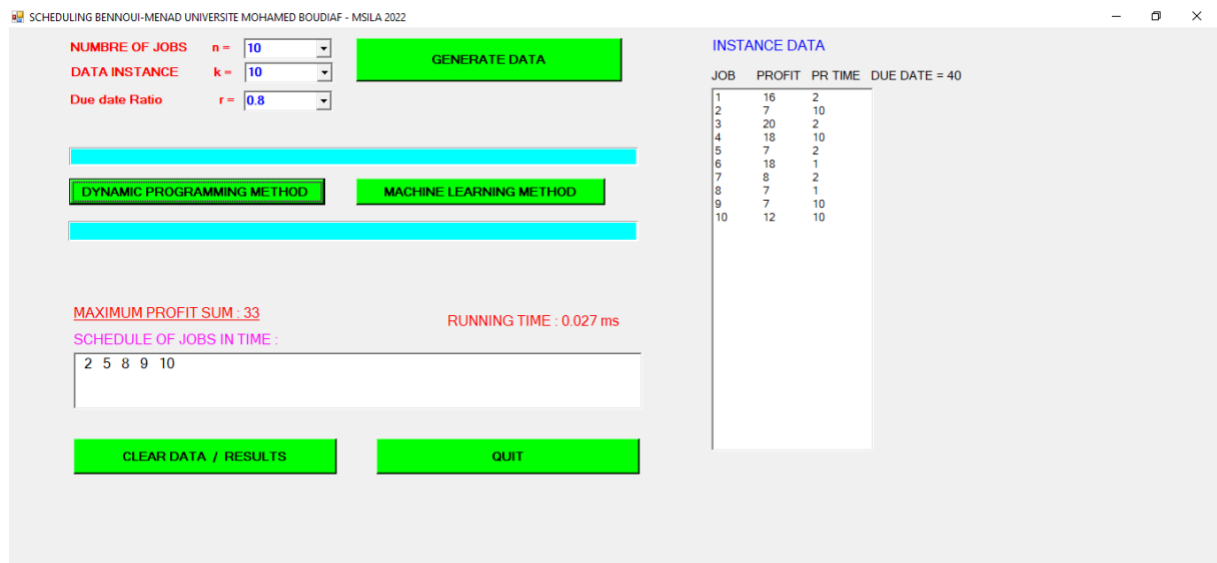


Figure 5.14: result of dynamic programming, number jobs, duo date ratio, $n=10$. $r=0.8$.

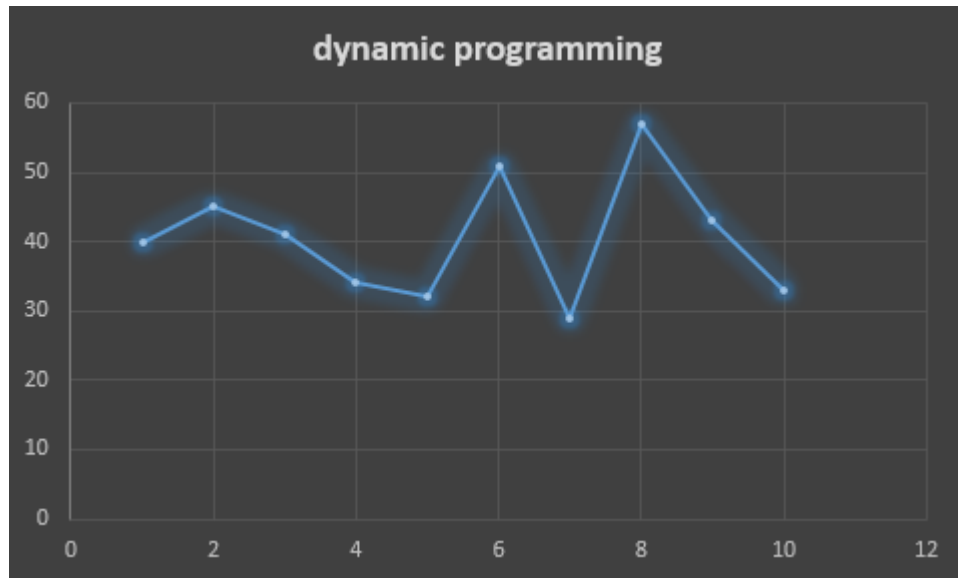


Figure 5.15:Graphic curve for dynamic programming, number jobs, duo date ratio, $n=10$. $r=0.8$.

Instance	Maximum Profit Sum
1	4117
2	3877
3	3890
4	3898
5	3982
6	3965
7	3962
8	4104
9	3991
10	4069
Average	3985.5

Tab 5.6. Dynamics programming.

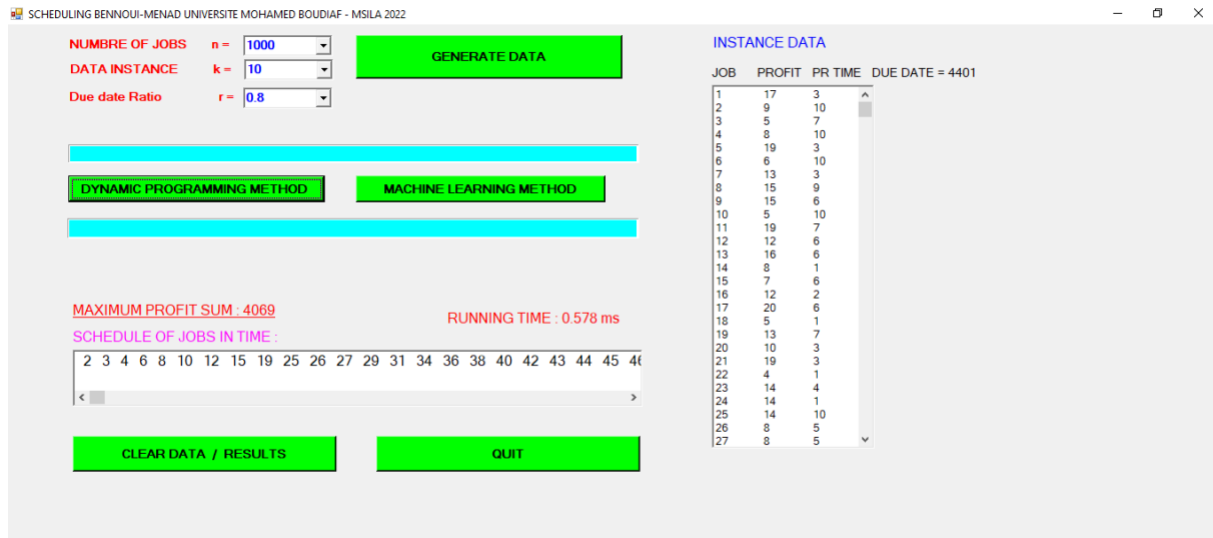


Figure 5.16. Result of dynamic programming, number jobs, duo date ratio, $n=1000$. $r=0.8$.

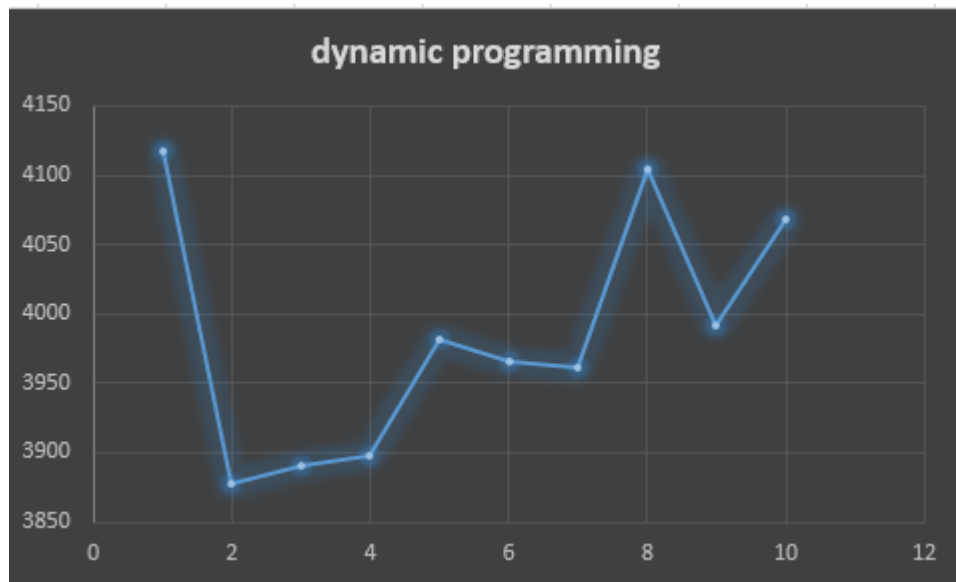


Figure 5.17. Graphic curve for dynamic programming, number jobs, duo date ratio, $n=1000$. $r=0.8$.

6. Machine Learning Steps:

The task of imparting intelligence to machines seems daunting and impossible. But it is actually really easy. It can be broken down into 7 major steps :

6.1. Collecting Data:

As we know, machines initially learn from the data that you give them. It is of the utmost importance to collect reliable data so that your machine learning model can find the correct patterns. The quality of the data that you feed to the machine will determine how accurate your model is. If you have incorrect or outdated data, you will have wrong outcomes or predictions which are not relevant.

Make sure you use data from a reliable source, as it will directly affect the outcome of your model. Good data is relevant, contains very few missing and repeated values, and has a good representation of the various subcategories/classes present.

6.2.Preparing the Data:

After you have your data, you have to prepare it. You can do this by :

Putting together all the data you have and randomizing it. This helps make sure that data is evenly distributed, and the ordering does not affect the learning process.

Cleaning the data to remove unwanted data, missing values, rows, and columns, duplicate values, data type conversion, etc. You might even have to restructure the dataset and change the rows and columns or index of rows and columns.

Visualize the data to understand how it is structured and understand the relationship between various variables and classes present.

Splitting the cleaned data into two sets - a training set and a testing set. The training set is the set your model learns from. A testing set is used to check the accuracy of your model after training.

6.3.Choosing a Model:

A machine-learning model determines the output you get after running a machine-learning algorithm on the collected data. It is important to choose a model, which is relevant to the task at hand. Over the years, scientists and engineers developed various models suited for different tasks like speech recognition, image recognition, prediction, etc. Apart from this, you also have to see if your model is suited for numerical or categorical data and choose accordingly.

6.4.Training the Model:

Training is the most important step in machine learning. In training, you pass the prepared data to your machine-learning model to find patterns and make predictions. It results in the model learning from the data so that it can accomplish the task set. Over time, with training, the model gets better at predicting.

6.5.Evaluating the Model:

After training your model, you have to check to see how it's performing. This is done by testing the performance of the model on previously unseen data. The unseen data used is the testing set that you split our data into earlier. If testing was done on the same data which is used for training, you will not get an accurate measure, as the model is already used to the data, and finds the same patterns in it, as it previously did. This will give you disproportionately high accuracy.

When used on testing data, you get an accurate measure of how your model will perform and its speed.

6.6.Parameter Tuning:

Once you have created and evaluated your model, see if its accuracy can be improved in any way. This is done by tuning the parameters present in your model. Parameters are the variables in the model that the programmer generally decides. At a particular value of your parameter, the accuracy will be the maximum. Parameter tuning refers to finding these values.

6.7.Making Predictions :

In the end, you can use your model on unseen data to make predictions accurately.

Steps of Dynamic Programming Approach

Dynamic Programming algorithm is designed using the following four steps –

Characterize the structure of an optimal solution.

Recursively define the value of an optimal solution.

Compute the value of an optimal solution, typically in a bottom-up fashion.

Construct an optimal solution from the computed information.

7. Analysis of results

We note in the test of machine learning and dynamic programming that the dynamic programming algorithm got very good results and exact results but took longer compared to machine learning. we notice that machine learning obtained very good results sometimes it is not accurate and took less time compared to dynamic programming machine learning. It is easy to implement and can be parallelized easily. A priority implementation makes use of large item set properties. It is one of the most interpretable machine learning algorithms, making it easy to explain to others. It is easy to use as it requires minimal tuning. It is the mostly widely used machine learning technique that runs fast. Easier to inspect and less complex.

8. Conclusion

In this chapter we have presented our implementation, then we have presented the hardware and software environment to implement, after that we compared the machine learning algorithm and dynamic programming after we compared the number of processors in the uniquely, and finally we discussed the results obtained.

GENERAL CONCLUSION

The main goal of our work was to apply machine-learning algorithms to solve the single machine-scheduling problem, which is an extremely complex problem. It is classified as combinatorial problems.

We created an application to fix this problem with the use of machine learning and dynamic programming algorithms, and the more we add the only to compare between machine learning and dynamic programming algorithms, also we try to see who is best to maximizer the Makespan. Finally, we deduce that machine learning and dynamic programming algorithms are the best to maximize the Makespan quickly..

In the first chapter, we studied machine learning algorithms and their principles, as well as their mechanism of operation.

The second chapter, we studied the sequencing in general cases, so we present and their models.

In the third chapter, we presented what the others are doing in our theme.

In the fourth chapter, we explained our help to solve the single machine problem with the use of machine learning and dynamic programming algorithms.

Finally, we presented our application and some explanation for the used, also the parameters of our application.

BIBLIOGRAPHY

[1] Aytug, H., Bhattacharyya, S., Koehler, G. J., and Snowdon, J.L., "A Review of Machine Learning in Scheduling," IEEE Transactions on Engineering Management, Vol.41, No. 2, 165-171 (1994).

[2] Baker, K.R., «Sequing Rules and Due-Date Assistments in a Job Shop», Management Science, Vol. 30, No. 9, 1093-1104 (1984)

[3] Michalski, R.S., "A Theory and Methodology of Inductive Learning," in Machine Learning, edited by R. Michalski, J. Carbonell and T. Mitchell, Tioga, Palo Alto, CA (1983).

[4] Sas.com. 2022. A guide to the types of machine learning algorithms.
<https://www.sas.com/en_gb/insights/articles/analytics/machine-learning-algorithms.html>
By Katrina Wakefield, SAS UK

[5]García, D., 2022. *Manufacturing Systems Scheduling through Machine Learning*.
<https://www.academia.edu/60859838/Manufacturing_Systems_Scheduling_through_Machine_Learning>

[6] Steckel, K.E. and Solberg, J.J., "Loading and control policies for a flexible manufacturing system', International Journal of Production Research, Vol. 19, N. 5 481-490 (1981)

[7] <<https://www.simplilearn.com/10-algorithms-machine-learning-engineers-need-to-know-article>>

[8] Goldberg, D. E. (1989), "Genetic Algorithms in Search, Optimization & Machine Learning", Reading: Addison-Wesley, New York: Addison-Wesley, Bibcode: 1989gaso.book...G, ISBN 978-0-201-15767-3

[9]<https://us.hudson.com/legal/blog/postid/513/predictive-artificial-intelligence-science-fiction-e-discovery-truth>

- [10] F. Sahel, Etude de performance des règles de priorité dans la résolution de problème d'ordonnancement job shop, mémoire de fin d'étude Master ,Université Aboubakr Belkaïd-Tlemcen,2020/2021
- [11] H. Boukef, Sur l'ordonnancement d'ateliers job-shop flexibles et flow-shop en industries pharmaceutiques Optimisation par algorithmes génétiques et essais particuliers, thèse de Doctorat, Université de Tunis el Manar École National D'ingénieur de Tunis, 2009.
- [12] A.KACIMI EL HASSANI H.ABASSI, L'optimisation à base de simulation pour les chaînes logistiques, mémoire de fin d'étude Master, Université Mohamed Boudiaf-Msila.2019/2020.
- [13] A. HANNACHE et A. LEMMOUIS. Résolution d'un problème d'ordonnancement de type job Shop avec contrainte de transport, mémoire de fin d'étude Master, UniversitéAbou Berk Belkaïd-Tlemcen ,2016.
- [14] <<https://www.geeksforgeeks.org/maximize-binary-matrix-flipping-submatrix/>>

Abstract

Machine learning approach for single machine scheduling problems This theme aim to design a machine-learning algorithm to tackle a NP-hard single machine scheduling problems with big size. This approach consists on two parts: the first part “learning” aims to learn the system by supplying the system with a significant number of small size instances solved by an exact method as dynamic programming. The second step consist to design a machine learning approach to tackle some big size instances, as we go along, even these instances supply our system to improve its efficiently. A comparison with chosen metaheuristic is needed to justify the contribution.

Key words: Machine Learning; Single Machine Scheduling; Optimization; Dynamic Programming.

Résumé

Approche d'apprentissage automatique pour les problèmes de planification de machine unique Ce thème vise à concevoir un algorithme d'apprentissage automatique pour aborder un NP-dur problèmes de planification de machine unique avec grande taille. Cette approche se compose de deux parties : les premières par « apprentissage » visent à apprendre le système en fournissant au système un nombre significatif d'instances de petite taille résolues par une méthode exacte de programmation dynamique. La deuxième étape consiste à concevoir une approche d'apprentissage automatique pour aborder certaines instances de grande taille, au fur et à mesure que nous avançons, même ces instances fournissent notre système pour l'améliorer efficacement. Une comparaison avec la métaheuristique choisie est nécessaire pour justifier la contribution.

Mots-clés : Machine Learning; Single Machine Scheduling; Optimisation; Programmation dynamique.

الملخص

نهج التعلم الآلي لمشاكل جدولة الآلة الواحدة يهدف هذا الموضوع إلى تصميم خوارزمية التعلم الآلي لمعالجة مشاكل جدولة الآلة المفردة ذات الحجم الكبير. يتكون هذا النهج من جزأين: تهدف الخطوة الأولى «التعلم» إلى تعلم النظام من خلال تزويد النظام بعدد كبير من حالات الحجم الصغير التي يتم حلها بطريقة دقيقة مثل البرمجة الديناميكية. تتمثل الخطوة الثانية في تصميم نهج التعلم الآلي لمعالجة بعض الحالات ذات الحجم الكبير، مع تقدمنا، حتى هذه الحالات تزود نظامنا بتحسينه بكفاءة. هناك حاجة إلى مقارنة مع ميتاهويستي المختارة لتبرير المساهمة.

الكلمات الرئيسية: التعلم الآلي ؛ جدولة آلة واحدة ؛ التحسين الأمثل ؛ البرمجة ديناميكية