UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

Escuela de Ciencias Matemáticas y Computacionales

TÍTULO: Discrete Event Simulation Using High Performance Computing for Manufacturing Processes

Trabajo de integración curricular presentado como requisito para la obtención del título de Ingeniero en Tecnologías de la Información.

Autor/a:

Murillo Lucero Luis Eduardo

Tutor/a:

Armas Andrade Rolando, Ph.D.

Cotutor/a:

Pineda Israel, Ph.D.

Urcuquí, 02 de Mayo de 2023

Autoría

Yo, **Luis Eduardo Murillo Lucero**, con cédula de identidad 1724738826, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el/la autor/a del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, 02 de Mayo de 2023.

Firmado electrónicamente por:
LUIS EDUARDO
MURILLO LUCERO

Luis Eduardo Murillo Lucero

CI: 1724738826

Autorización de publicación

Dedication

A quienes me han apoyado en este proceso. Primeramente agradezco a Dios y a mis padres Lucia y Eduardo ya que sin su ayuda constante no hubiera podido completar este logro. También quiero dedicar este logro a Isaac, Luciana y Paola quienes son mi familia cercana. Además quisiera agradecer a mis profesores que durante tantos años me compartieron su conocimiento. Agradezco a mi alma mater Yachay por proveerme de diferentes habilidades y enseñanzas a lo largo de este tiempo. Agradezco a mis amigos que fueron una familia para mi en varias etapas durante la universidad, Stalyn, Evelyn, Mayra, Julian, Mateo, Sherald, Karen y disculpen si me olvido de alguien. Finalmente, quisiera dedicar este logro a quienes se han fijado en mí como un ejemplo a seguir. Les incentivo a perseguir sus sueños y metas, sin importar lo que otras personas crean primero crean en ustedes.

To those who supported me in this process. First of all, I thank God and my parents Lucia and Eduardo because, without their constant support, I would not have completed this achievement. Besides, I want to dedicate this achievement to Isaac, Luciana and Paola who are my close family. I would also like to thank my professors, who have shared their knowledge with me for many years. I thank my alma mater Yachay for providing me with different skills throughout this time. I thank my friends who were family to me at various stages during college, Stalyn, Evelyn, Mayra, Julian, Mateo, Sherald, Karen and sorry if I forgot anyone. Finally, I dedicate this achievement to those who have looked up to me as an example. I encourage you to pursue your dreams and goals, no matter what. Over other people's thoughts, believe in yourself first.

Luis Eduardo Murillo Lucero

Resumen

El objetivo de este trabajo es implementar una herramienta basada en simulación de eventos discretos, cuya finalidad sea la organización y optimización de recursos para asuntos de manufactura industrial. Este trabajo proporciona una solución basada en simulación al problema de programación del taller de trabajo y analiza la eficiencia en sistemas de cómputo de alto rendimiento. El problema mencionado busca potenciar el mejor arreglo de tareas para mejorar tiempos y optimizar recursos. Es considerado un problema NP-Completo por la cantidad de variables a evaluar, de ahí su complejidad e importancia para la planeación industrial. En este estudio se realizó una simulación serial en base a eventos discretos, mismo que se probó con tres diferentes conjuntos de datos ft06, ft10 y ft20 propuestos para este tipo de enfoques. Este método es comparado con la mejor herramienta de optimización en la actualidad, OR-Tools, desarrollada por Google. Finalmente, la simulación es implementada en su versión serial y paralela para incluirlo en un sistema de cómputo de alto rendimiento, colectando la información de las métricas computacionales durante el proceso con la finalidad de buscar alternativas para futuras implementaciones con modelos más robustos y relacionados a escenarios reales. Los resultados mostraron una implementación en base a simulación de eventos discretos sostenible y escalable para modelos más complejos, además la misma se puede aplicar en sistemas de cómputo de alto rendimiento.

Palabras Clave:
Simulación por Eventos Discretos, Computo de Alto Rendimiento, OR-TOOLs, Optimización, Manofactura.

Abstract

This work aims to implement a tool based on the discrete event simulation, whose purpose is to organize and optimize resources for industrial manufacturing issues. This work provides a simulation-based solution to the job shop scheduling problem and analyzes the efficiency of high-performance computing systems. The problem mentioned above seeks to promote the best arrangement of tasks to improve time and optimize resources. It is considered an NP-Completeness problem due to the number of variables to be evaluated, hence its complexity and importance for industrial planning. In this study, the serial simulation was developed based on discrete events and tested with three different data sets for this type of approach, ft06, ft10, and ft20. This method is compared to the best optimization tool currently, OR-Tools, developed by Google. Finally, the simulation is implemented in serial and parallel versions to include it in a high-performance computing system. It collects information on the computational metrics to find alternatives for future implementations with more robust models and real scenarios. The results exhibited an implementation based on sustainable and scalable discrete event simulation for more complex models, and can also be implemented in high-performance computing systems.

Keywords:

Discrete Event Simulation, High-Performance Computing, OR-TOOLs, Optimization, Manufacturing.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Currently, the standard priority in our industrialized and interconnected world is using resources efficiently during manufacturing sequences. In the existing intensely dynamic industry, the manufacturing sector faces the relentless challenge of delivering creative goods on a shorter time-to-market basis. Thus, the integration of supplies and optimization during manufacturing directly impacts the performance of industries. Chryssolouris [10] states that the industrial and technological development of a country is correlated with the increase of machine tools and optimization in the manufacturing sector. Hence, the relevance of focusing on this kind of process, understanding that manufacture is the conversion of raw materials and information into items to satisfy human needs becoming a critical topic to analyze due to its influence on our daily life.

On the other hand, product requirements are becoming more complex and customized. Thus, the industries face a new production challenge, especially non-automated industries. Optimization and automation became key points to achieve even ambitious goals such as the industry 4.0 [11] hence the importance of seeking technological approaches to obtain results. The complexity of the problem increases because of the flexibility in the production process, especially in the scheduling area. Thus, finding the best optimization for increasingly dynamic and flexible processes is hugely demanding. In brief, finding the best or the nearest arrangement for the process is a complex and expensive problem for manufacturing systems. In this context, knowing that production is the main business operation of companies. As a result, moving ahead with new technologies is a crucial point in their value chain network [12]. Usually, the ideal industry model avoids the maximum of errors and inefficient steps during the production line because errors imply cost and waste of time for the entire entity. Consequently, researchers such as Chryssolorious [10] and Zhou [12] claim to improve the IT tools and methods for total quality management during the process.

Notably, the industrial trend has been technological implementation and improvement during the last decade. The term "Industry 4.0" [13] [14] is hugely popular due to the giant inversion in "Smart Factory," a term used to express production processes that have included disruptive technology. To summarise, Industry 4.0 is the fourth industrial revolution which means the next era of the industry implementing groundbreaking technology in the current system, which is identified by the use of electronics and information technology to automate production; the concept of Industry 4.0, Smart Factory and Smart Cities [15]. The massive information and digitalization generated by industries during their process carry this technological phenomenon. Industry 4.0 hopes to revitalize the automation industry-supported but is not limited by new technologies such as IoT, Artificial Intelligence, machine learning, Simulation, and other trends.

Above all disruptive technologies, Simulation, in particular, has especially applied to perform manufacturing advances. The relationship between Simulation and manufacturing has risen in recent years because of the growing interest of academia and industry in implementing *digital twins* [16]. The expression "digital twin" (DT) means a digital representation of the physical complex production model, in the case of manufacture. It is a controlled digitalized mirror of reality. Digital twinning allows a low-cost testing environment founded on Simulation to analyze different approaches and improvements. Under those circumstances, Simulation became a trending matter, especially to achieve the next automated step and optimization. Hence, this work focuses on the Discrete Event Simulation procedure and its connection with digital manufacturing twins. The benefits and concepts of implementing these intelligent systems are explained in the Chapter 2.

Finally, It is important to emphasize the enormous interest of big tech companies in this kind of technology. A great example is the technological giant NVIDIA and its incredible effort to spread manufacturing Simulation. NVIDIA created a revolutionary virtual platform built for cooperation and real-time photorealistic Simulation called omniverse. The concept is to use Simulation to test different production lines before implementing them in real life[1]. NVIDIA omniverse for manufacture is a clear example of the importance of disruptive technologies applied to current problems. Then, the new direction of this virtual platform at the end of the day is advanced Simulation to implicate people in the new atmosphere. In short, Simulation is the stakeholder of this type of advancement, showing the significance of its introspection and application to solve real-world problems.

1.2 Problem Statement

The inspiration to develop this work lies in the lack of technological transformation in the manufacturing sector, especially in developing regions. The local industrial sector competes with thousands of industries simultaneously due to globalization. It implies improving products and optimizing processes to be competitive. The main difference between huge and growing companies arose today in technological implementation and innovation. The lack of disruptive technology guides companies to have less profit. Thus the question becomes: How can industries leverage this problem?

According to the Economic Commission for Latin America and the Caribbean (ECLAC), a United Nations entity, the region does not show a significant association between services value-added and manufacturing competitiveness [17]. On the other hand, countries moving to technological transformation, such as the Association of Southeast Asian Nations (ASEAN) countries, show a significant positive relationship between their participation in foreign markets and their level of servicification, particularly for foreign value-added. Indeed, ECLAC expects 50% employment growth in the manufacturing sector by 2030 promoting social inclusion and economic recovery. Therefore, today manufacturing directly impacts the economy of many regions. Just in Latin America and the Caribbean Region, manufacturing was 15.3% of the total value added of GDP in 2021 [18]. Besides, there are more complex cases; for example, in Ecuador, manufacturing influences 17.9% of the total value added of GDP at current prices in 2021.

Finally, ECLAC agrees that early signs of digitalization in traditional manufacturing

[1]Source: https://www.nvidia.com/en-us/omniverse/manufacturing/

exemplify business investment trends [19]. One, in particular, is optimizing manufacturing processes that are a significant financial benefit to companies owing to the importance of participating in data markets to accumulate and exchange data. Given these points, it is important to remark on the importance of the manufacturing industry and the close correlation between technology and innovation with the added value of a product.

1.3 Objectives

1.3.1 General Objective

Implement a manufacturing software tool based on discrete event simulation then compare the computational metrics using high-performance computing.

1.3.2 Specific Objectives

- Define an approximation to the manufacturing process scenario using discrete event simulation returning values consistent with manufacturing production requirements.
- Extend the results of the simulation returning a more visual representation of the process using gantt charts.
- Compare two different implementations—one using High-Performance Computing, and the other using the conventional approach.

Chapter 2

Theoretical Framework

This chapter defines useful notions and definitions, beginning with computer simulation and a high-performance overview of the complex concepts used in this work. This section matches the general concepts and their relations to propose a solution to the problem. Each of the concepts is paired with its corresponding graphical and mathematical representations.

2.1 Computer Simulation

Computer Simulation is the generic name of computer software based on the Simulation guide to manufacturing operations [10]. Simulation is defined as the imitation of real-world processes or systems over time [20]. In concept, a computer simulator uses decision variables as inputs. Decision variables are the design specifications, the workload, and the operational policies of a manufacturing system. The simulator assembles the inputs into a model of the manufacturing system. The model includes rules to guide the interactions between components in the system. The simulator input also includes the initial state of the system. Under the circumstances, the simulator follows the operations of the model, tracking events over time. As a result, the simulator output refers to performance measures which are a set of statistical values to be evaluated, e.g., the average number of parts in the system over time. Figure 2.1 details the basic simulation model, including its inputs and outputs.

Simulation is defined as an analyst tool because potential changes to the system can first be simulated. Therefore, before building manufacturing systems, a good practice is to

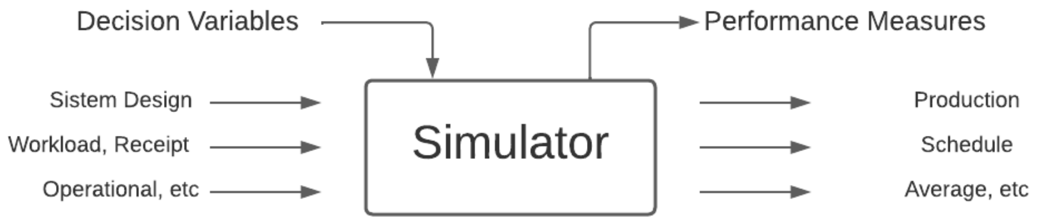


Figure 2.1: Elements of Computer Simulation

study the design in a simulation. Usually, the Simulation performs including other elements such as stochastic methods to reduce noise [21][22]. Another relevant improvement is using graphical tools such as Simulink to improve the visualization of the initial parameters [23]. Besides, Simulation in recent years includes heuristic approaches to obtain better results, such as genetic algorithms [24]. This approach has been studied for many years [25], but recently using the computational power. Most manufacturing simulations evolve with changing variables over time. It implies tracking state variables at separate points in a period. The points in time mean that an event occurs; an event is an instantaneous occurrence that may change the system's state. Usually, this kind of model is called a discrete event simulation model.

2.1.1 Discrete Event Simulation (DES)

The Discrete Event Simulation refers to a system modeling a long time that consists of state changes when an event occurs at discrete points in time [20]. Figure 2.2 illustrates the events (E_i) during a simulation along the period. E_i represents the state of the simulation (i.e., the plant) in a given moment. This illustration clarifies that discrete event simulation uses a non-constant time step, which reduces computational expenditure. The non-constant time increment Δt avoids long running times during periods when no meaningful activity is executed. The model proceeds by producing a sequence of system snapshots. The system snapshots are the representations of the evolution of the system through time. At a specific time t, the DES system is smart enough to list all activities currently in progress, when each such activity ends, and the system's state at this point. Finally, the statistical accumulated values and the states return at the end of the simulation. Besides, the DES models usually are connected with new disruptive technologies such as Digital Twins [26].

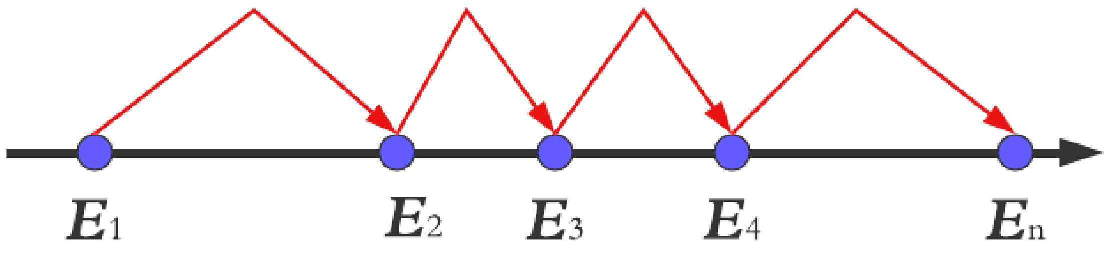


Figure 2.2: Discrete Event Simulation along time.

All discrete event-driven simulation models share the following components [10]:

- **System state**: The collection of state variables necessary to describe the system at a particular time.
- **System clock**: A variable giving the current value of simulated time.
- **Event list**: A list containing the next time when each type of event will occur.
- **Statistical counters**: Variables used for storing statistical information about system performance.
- **Initialization routine**: A subprogram to initialize the simulation model at time zero.
- **Timing routine**: A subprogram that determines the next event from the event list and then advances the simulation clock to the time when that event is to occur.
- **Event routine**: A subprogram that updates the system state when a particular type of event occurs (there is one event routine for each event type).
- **Library routines**: A set of subprograms used to generate samples from probability distributions that were determined as part of the simulation model.

- **Report generator**: A subprogram that computes estimates (from the statistical counters) of the desired measures of performance and produces a report when the simulation ends.
- **Main program**: A subprogram that invokes the timing routine to determine the next event and then transfers control to the corresponding event routine to update the system state appropriately. The main program may also check for termination and invoke the report generator when the simulation is over.

The Chapter 4 details more about the programming implementation and flow of the process in a Discrete Event Simulation System. Besides, Figure 2.3 details the common scheme in the Discrete Event Simulation Algorithms.

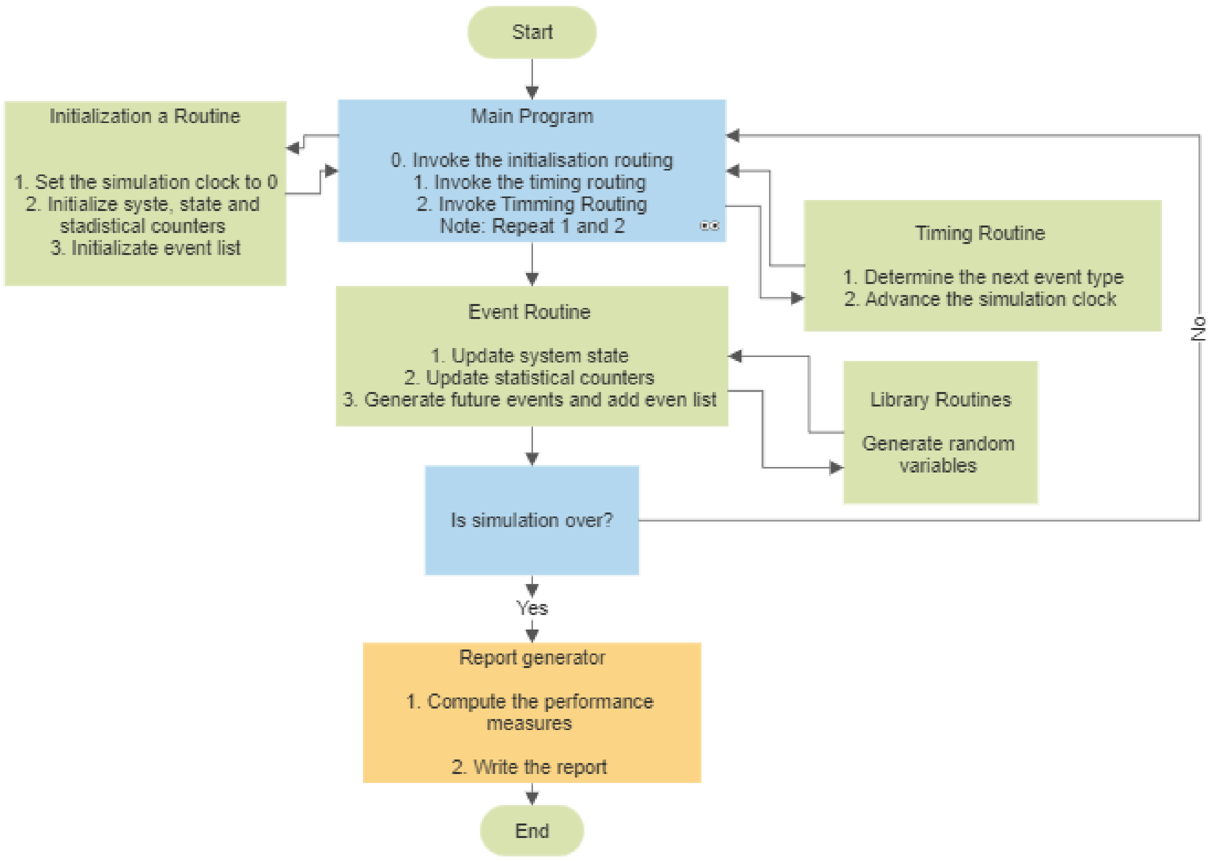

Start
Initialization a Routine
1. Set the simulation clock to 0
2. Initialize syste, state and stadistical counters
3. Initializate event list
Main Program
0. Invoke the initialisation routing
1. Invoke the timing routing
2. Invoke Timming Routing
Note: Repeat 1 and 2
Timing Routine
1. Determine the next event type
2. Advance the simulation clock
Event Routine
1. Update system state
2. Update statistical counters
3. Generate future events and add even list
Library Routines
Generate random variables
Is simulation over?
No
Yes
Report generator
1. Compute the performance measures
2. Write the report
End

Figure 2.3: Discrete Event Simulation Implementation Scheme.

2.1.2 Digital Twin (DT)

Digital Twin is software-based abstractions of complex physical systems connected to the real system via a communication link to continuously exchange data with the real environment and establish a dynamic digital mirror with a constantly running modeling engine [2]. Figure 2.4 shows the leading technologies and applications for Digital Twins. The figure illustrates the different layers explained later in this work and the potential technologies around DT. Digital Twin technology has evolved since 2022. Thus, DT is the convergence of different technologies,i.e., machine learning algorithms, data analysis, super-resolution visualization, statistical modeling, and Simulation. Primarily, the DT technology guides the manufacturing process to optimization. The mailable of DT allows imitating and molding physical manufacturing layers. It implies quick testing, allowing analyzing results before any physical change. Although Digital Visualizations, usually in 3D, are a constant improvement in DTs, it is just the first layer of all the technology. Other implementation layers include other fields and technologies. According to Jeong, DT technology has five implementation layers. The first one is Digital virtualization which refers to the digital representation and objectification process of components making up the real target world, such as machines, materials, people, and spaces. The second is DT synchronization, which implies real-time mutual communication between real-world and virtual-world components. The third layer is Modeling and Simulation, which analyzes and predicts the real world by simulating changes in virtual twin objects according to conditional changes. The fourth layer is Federated DT, which connects mutual federation and collaboration between DTs. The last is Intelligent DT services that manage service lifecycles based on intelligent and autonomous technologies and related platforms [27]. During this work, we focus on the third stage, ”Modeling and Simulation.” We use Simulation as a tool to improve manufacturing systems and analyze their relation with other technologies.

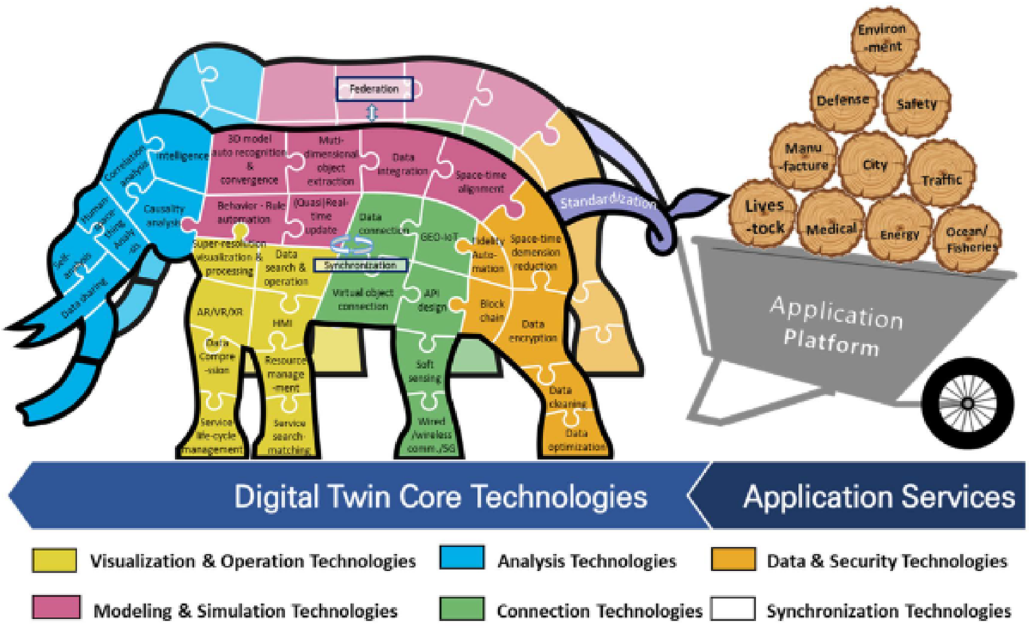


Figure 2.4: Digital twin essential technologies and application services. Source [2]

2.2 Problem Definition

2.2.1 Job Shop Problem (JSP)

The Job Shop Problem [28] known as JSP is defined by a finite set J of n jobs and a finite set M of m machines. Figure 2.5 describes the Job Shop Problem graphically. Figure describes three jobs, $J1$, $J2$, and $J3$, are scheduled on three machines, $M1$, $M2$, and $M3$. The graph at the top represents the precedence restrictions. The Gantt Chart at the bottom shows a viable timetable that adheres to the priority restrictions. We will refer to an nxm problem for convenience. The list $\left(\sigma_1^j, ..., \sigma_h^j, ..., \sigma_m^j\right)$ of the machines reflects the processing order though machines for each job $j \in J$. The term σ_h^j is called the $h-thoperation$ of job j and the term σ_m^j refers to the last *operation* of the job j. The non-negative integer p_{ij} represents the progressing time of the job j in the machine i. Each machine may only process one work at a time, and once a job begins, it must finish processing on that machine without interruption. It is advantageous because 1) each operation can be assigned to a machine, 2) machines can be evenly loaded 3) machine breakdowns can be accommodated easily. The common objective is to arrange a timetable of J on M that minimizes the makespan, i.e., the maximum realization time of the last procedure of any job in J. Makespan minimization for the Job Shop Problem is proved to be an NP-Hard problem for more than three tasks in two machines [29].

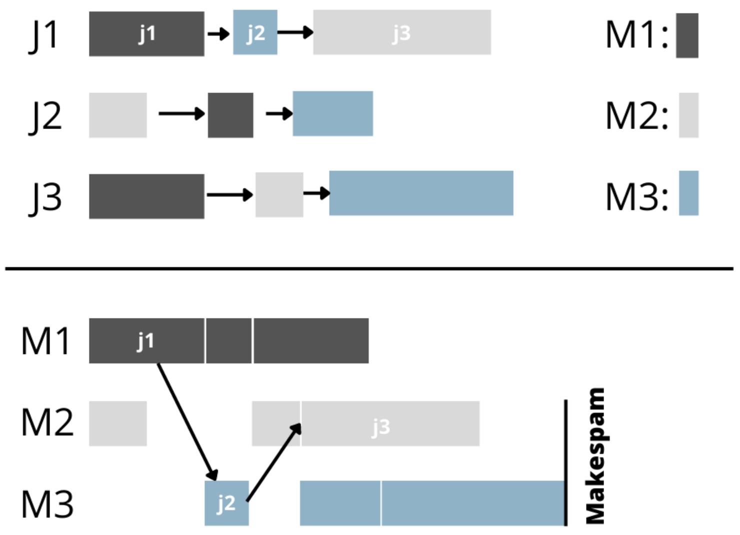


Figure 2.5: Job Shop Scheduling Problem and Gantt Chart distribution.

Gantt Chart

Gantt Chart is an analog representation of schedule [30]. It is widely used to represent solutions related to the Job Shop Problem. Figure 2.6 helps visualize Gantt Chart as a schedule detailing tasks, elements, and resources. The advantage of the Gantt Chart lies in analyzing the geometric relation of its elements. Besides, Gantt Chart is a simple way to illustrate all the process time in one element. Thus, Gantt Chart serves to aid in measuring performance and comparing different arrangements of resources over time. In brief, Gantt Chart helps to analyze and visually compare different scheduling.

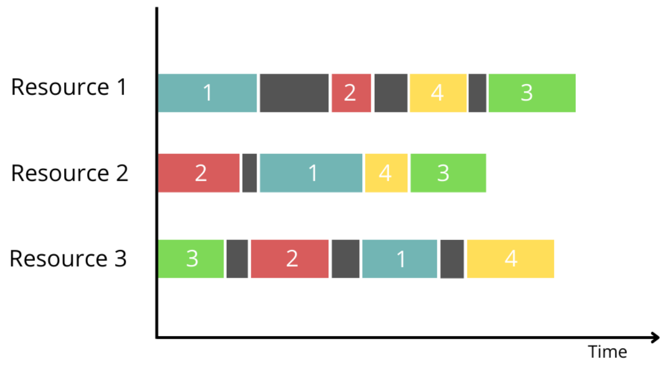


Figure 2.6: A Gantt Chart.

2.2.2 NP-hardness

NP-hard are optimization problems that can not be solved by a Non-deterministic Turing Machine in polynomial time. Optimization problems as difficult as the manufacturing scheduling or even more difficult are called NP-hard Problems [31]. Usually, NP-hard problems relate to decision-making problems. The problem Π is "NP-hard" since it is at least as hard as NP-complete problems. It implies that it can not be solved in polynomial time unless P=NP [32]. Notably, the problem definition depends on the complexity. In this particular case, complexity refers to the computational effort required by a solution algorithm. Figure 2.7 illustrates the family of P, NP, NP-complete, and NP-hard problems in a diagram of Euler. It is the most famous and current representation of this kind of problem. Thus, the illustration considers the possibility that $P = NP$ or $P \neq NP$. To analyze the problem in detail and understand its importance, we define the following concepts:

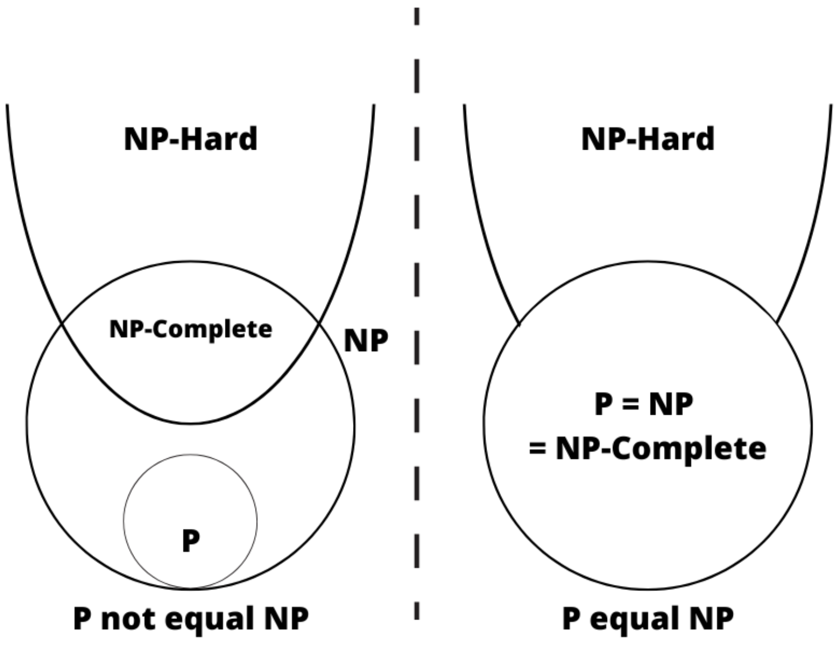


Figure 2.7: Euler diagram for P, NP, NP-complete, and NP-hard set of problems.

Deterministic Turing Machine (DTM)

In order to formalize the notion of algorithm it is necessary to define a particular computational model. The deterministic one-tape Turing machine (DTM) [33] is illustrated schematically in Figure 2.8. It consist in a finite state control, a read-write head, and a tape made up of a two-way infinite sequence of tape squares, i.e. . . . , -2, -1,0,1,2,...

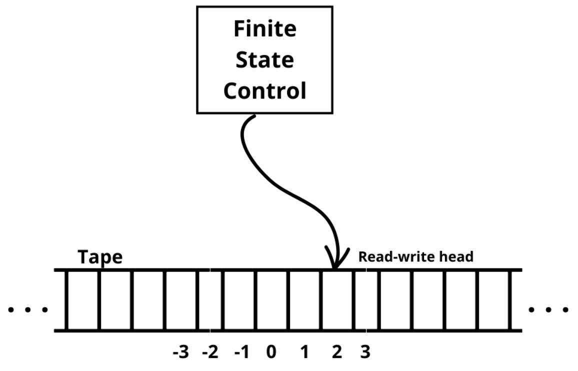


Figure 2.8: Schematic representation of deterministic one-tape Turing machine.

A program for a DTM specifies the following terms:

- A finite set Γ of tape symbols, including the subset $\Sigma \subset \Gamma$ of input symbols and a distinguished blank symbol $b \in \Sigma - \Gamma$;
- a finite set Q of states, including a distinguished state-start q_0 and two halt-states q_Υ and q_N;
- a transition function $\delta : (Q - (q_\Upsilon, q_N)) \times \Gamma \to Q \times \Gamma \times \{-1, +1\}$

The process of a program is straightforward. The input is a string $x \in \sum^*$ placed in the tape squares 1 through $|x|$, all the other squares contain the blank symbol b. The q_0 is the initial state where the program starts with the read-write head scanning tape square 1. The procedure is in a step-by-step method. The current state in the system is q which belongs to $Q - \left\{q_\gamma, q_N\right\}$, some symbol $s \in \Gamma$ is the tape square being scanned, and the value of $\delta\left(q, s\right)$ is defined. Suppose $\delta\left(q, s\right) = (q', s', \Delta)$. It implies that the read-write head erases s and writes s' instead. The process finishes if the state q equals q_γ; otherwise, the process continues. If the move is to the left, it implies $\Delta = -1$; on the other hand, if the move is to the right, it means $\Delta = +1$. In the same way, the state q changes to q', which completes one *step* of the computation, ready to proceed to the next step, if there is one more.

In general, the program M with input Σ allows $x \in \Sigma^*$ if and only if M halts in condition q_γ when involved to input x. The language L_M admitted by the program M is provided by:

$$L_M = [x \in \Sigma^*; M \ \ accepts \ \ x]$$

The DTM program M solves the decision problem Π under encoding scheme e if M halts for all input strings over its input alphabet and $L_M = L[\Pi, e]$. For all the inputs $x \in \Sigma^*$, its *time complexity function* $T_M : Z^+ \to Z^+$ is given by:

$$T_M(n) = max\{m : x \in \Sigma^*, |x| = n, s.t., M \ \ on \ \ input \ \ x \ \ takes \ \ time \ \ m\}$$

The program M is *polynomial time DTM program* if there exists a polynomial p such that, for all n^+, $T_M(n) \leq p(n)$. Thus, now let give the formal definition of the class P as follows:

$$P = \{L : there \ \ is \ \ a \ \ polynomial \ \ time \ \ DTM \ \ program \ \ M \ \ for \ \ which \ \ L = L_M\}$$

As a final point, a decision problem Π belongs to P under the encoding scheme e if $L[\Pi, e] \in P$. It means if there is a polynomial time DTM program that *solves* Π under the encoding scheme e.

Nondeterministic Turing Machine (NDTM)

The NDTM model has exactly the same structure as a DTM, except that it has a guessing module having its own write-only head as illustrated in Figure 2.9. The NDTM program uses the same principles as the DTM procedure explained previously. The transition function is $\delta : \left(Q - \left\{q_\gamma, q_N\right\}\right) \times \Gamma \to Q \times \Gamma \times \{-1, +1\}$ and keeps the tape alphabet Γ, input alphabet , blank symbol b, state set Q, initial state q_0, halt states q_γ and q_N. The NDTM differs in two stages: "guessing" and "checking" [32].

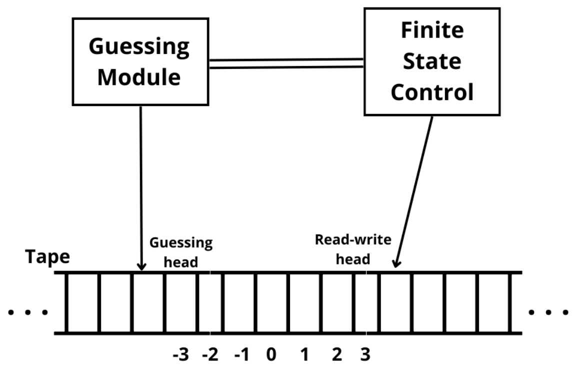


Figure 2.9: Schematic representation of nondeterministic one-tape Turing machine.

For the reason that NDTM program M will have an infinite number of possible computations for a given input string x, one for each guessing string Γ^*, M accepts x if at least one is an accepting computation. Then, the recognized language for M is:

$$L_M = [x \in \Sigma^*; M \ \ accepts \ \ x]$$

The NDTM program M seeks to be the minimum required time of the number of steps in the checking and guessing stages up until the stop in the state q_γ. The *time complexity function* $T_M : Z^+ \rightarrow Z^+$ for M is:

$$T_M(n) = max\left\{\{1\} \bigcup \{m : x \in L_M, |x| = n, s.t., time \ \ to \ \ accept \ \ x \ \ by \ \ M \ \ is \ \ m\}\right\}$$

The NDTM program M is a polynomial time program if there exist a polynomial p that fixes the conditions $T_M(n) \leq p(n)$ for all $n1$. Thus, the NP class is formally defined as follows:

$$NP = \{L : there \ \ is \ \ a \ \ polynomial \ \ time \ \ NDTM \ \ program \ \ M \ \ for \ \ which \ \ L_M = L\}$$

To sum up, a decision problem Π will belong to the NP class under the encoding scheme e if the language $L\left[\Pi, e\right] \in NP$.

Given these points, now we can define $NP\!-\!hard$ in formal terms. A string relation R is $NP\!-\!hard$ if there is some NP-complete language L such that $L \ \alpha_\Upsilon \ R$. The search problem Π is NP-hard if the string relation $R\left[\Pi, e\right]$ is NP-hard. In simple terms, a search problem

Π is NP-hard if there exists one NP-complete problem Π‘ that Turing reduces to Π. Thus, it is not difficult to imply that if a string relation R is $NP-hard$, then it does not have a polynomial solution time unless P=NP [31] (which is an unsolved mathematical problem). Under those circumstances, researchers propose models to approximate the problem to the manufacturing area. In the next section, we detail models related to Flexible JSP, and their main constraints seek to approximate the best solution for this difficult problem.

2.2.3 Disjunctive Model

In this stage, it is important to define the problem in mathematical terms. For this reason, we use the disjunctive model of the FJSP, originally proposed by Manne [34]. The model definition is presented in Table 2.1. The aim is to minimize the makespan function denoted such as C_{max}. Thus, the jobs are denoted as j and J as the set of all jobs. A specific machine is i and the set of all machines participating in the process is M. Then, X_{ij} represents an integer start time of a specific job j in a machine i. The symbol z is the decision variable, and V is a great enough value calculated as:

$$V = \sum_{j \in J} \sum_{i \in M} p_{ij}$$

Therefore, the model proposed by Manne includes six main constraints, which close the model to a real scenario. Mainly, the constraints avoid values outside a logical range. For example, the constraints do not allow to carry out operations without order. In real scenarios, the sequence of steps in manufacturing processes must follow the instructions to obtain the exact final product. The constraints that define the problem are the following:

- Constraint (1) dictates that the starting time of any job should be an integer greater than zero.
- Constraint (2) says that the starting time of a job should be greater or equal to the starting point of the previous job plus the processing time.
- Constraint (3) ensures that all the operations are carried on in the proper order.
- Constraint (4) ensures that no two jobs use the same machine simultaneously.

- Constraint (5) the makespan is greater than all the starting times plus its respective processing time for all jobs.
- Constraint (6) says that the decision variables are either zero or one.

$$\begin{array}{rll} min & C_{max} & \\ s.t & x_{ij} \geq 0 & \forall j \in J, i \in M\,(1) \\ & x_{\sigma_h^j, j} \geq x_{\sigma_{h-1}^j, j} + p_{\sigma_{h-1}^j, j}\text{t} & \forall j \in J, h = 2, ..., m\,(2) \\ & x_{ij} \geq x_{ik} + p_{ik} - V \cdot z_{ijk}, & \forall j, k \in J, jk, i \in M\,(3) \\ & x_{ik} \geq x_{ij} + p_{ij} - V \cdot \left(1 - z_{ijk}\right), & \forall j, k \in J, jk, i \in M\,(4) \\ & C_{max} \geq x_{\sigma_m^j, j} + p_{\sigma_m^j, j}\text{t} & \forall j \in J\,(5) \\ & z_{ijk} \in \{0, 1\} & \forall j, k \in J, ii \in M\,(6) \end{array}$$

$$min \quad C_{max} \tag{2.1}$$

$$s.t \quad x_{ij} \geq 0 \qquad \forall j \in J, i \in M\,(1) \tag{2.2}$$

$$x_{\sigma_h^j, j} \geq x_{\sigma_{h-1}^j, j} + p_{\sigma_{h-1}^j, j}\text{t} \qquad \forall j \in J, h = 2, ..., m\,(2) \tag{2.3}$$

$$x_{ij} \geq x_{ik} + p_{ik} - V \cdot z_{ijk}, \qquad \forall j, k \in J, jk, i \in M\,(3) \tag{2.4}$$

$$x_{ik} \geq x_{ij} + p_{ij} - V \cdot \left(1 - z_{ijk}\right), \qquad \forall j, k \in J, jk, i \in M\,(4) \tag{2.5}$$

$$C_{max} \geq x_{\sigma_m^j, j} + p_{\sigma_m^j, j}\text{t} \qquad \forall j \in J\,(5) \tag{2.6}$$

$$z_{ijk} \in \{0, 1\} \qquad \forall j, k \in J, ii \in M\,(6) \tag{2.7}$$

Table 2.1: Disjunctive Model of Manne

This sort of problem is usually approximated by Mixed Integer Programming (MIP) approaches. It represents a large part of the contributions of developing technologies in this area. In the section 3 there are more details about these contributions, and the latest research trends stretch different methods to enhance manufacturing optimization. In brief, diving into this problem with different strategies lies in its application in different industries to improve results and seek technological growth connecting manufacturing needs with novel solutions.

2.2.4 Disjunctive Model of Liao

The disjunctive model of Liao adds continuous, surplus variables q_{ijk} to constraints (3) and (4) [29]. The Table 2.2 shows the added constraints. Liao says that bounds on the variables are easier to handle than linear constraints. Liao introduces new decision variables that reduce the number of linear constraints, but it introduces variables and upper bounds. Liao stands that this transformation improves performance.

$$V \cdot z_{ijk} + \Big(x_{ij} - x_{ik}\Big) - p_{ik} = q_{ijk}, \qquad \forall j,k \in J, i \in M \quad (7)$$
$$q_{ijk} \leq V - p_{ij} - p_{ik}, \qquad \forall j,k \in J, i \in M \quad (8)$$

Table 2.2: Disjunctive Model of Liao

2.2.5 Time-indexed Model

The time- indexed model was originally proposed by Bowman. In this case, we describe the model of Kondili because the computational benefits compare with previous time-indexed proposals [29]. In the model the decision variable x_{ijk} is 1 if the job j stars at time t at the machine i. The Table 2.3 describes the time-indexed MIP model. The objective function is to minimize the C_{max} the same as the previous models.

The objective function is to minimize the C_{max} since it is important because the objective function will be the same during this work. The constraint (9) ensures that each machine contains exactly one job. The constraint (10) implies that the makespan is at least the largest completion time of the last operation of all jobs j. The constraint (11) ensures that the machine is not over-capacitated during the process. The last constraint (12) is the precedence constraint which implies that all job operations are executed in a specific order.

$$
\begin{aligned}
& min\, C_{max} \\
& s.t \sum_{t\in H} x_{ijt} = 1, && \forall j \in J, i \in M \,(9) \\
& \sum_{t\in H} \Big(t + p_{ij}\Big) \cdot x_{ijt} \leq C_{max}, && \forall j \in J, i \in M \,(10) \\
& \sum_{j\in J} \sum_{t'\in T_{ijt}} x_{ijt'} \leq 1, && \forall i \in M, t \in H, \\
& && where \;\; T_{ijt} = \Big\{t - p_{ij} + 1, ..., t\Big\} \,(11) \\
& \sum_{t\in H} \Big(t + p_{\sigma^j_{h-1},j}\Big) \cdot x_{\sigma^j_{h-1},jt} \leq \sum_{t\in H} t \cdot x_{\sigma^j_{h-1},jt}, && \forall j \in J, h = 2, ..., m \,(12) \\
& x_{ijt} \in \{0,1\} && \forall j \in J, i \in M, t \in H \,(13)
\end{aligned}
$$

Table 2.3: Time-indexed model

2.2.6 Rank-base model

The rank-based model was defined by Wagner [29]. The model defines the decision variables as follows: x_{ijk} is equal to 1 if the job j is scheduled at the $k-th$ position in the machine i and h_{ik} denotes the start time of the job in the machine i at the $k-th$ position. The Table 2.4 describes the rank-based model mathematically.

$$
\begin{aligned}
& min\, C_{max} \\
& s.t \sum_{j\in J} x_{ijk} = 1 && \forall i \in M, k = 1, ..., n \,(14) \\
& \sum_{k=1}^{n} x_{ijk} = 1, && \forall j \in J, i \in M \,(15) \\
& h_{ik} + \sum_{j\in J} p_{ij} x_{ijk} \leq h_{i,k+1}, && \forall i \in M, k = 1, ..., n \,(16) \\
& \sum_{i\in M} r_{ijl} h_{ik} + \sum_{i\in M} r_{ijl} p_{ij} \leq V \cdot \left(1 - \sum_{i\in M} r_{ijl} x_{ijk}\right) && \forall j \in J, i \in M, k, k' = 1, ..., n, \\
& + V \cdot \left(1 - \sum_{i\in M} r_{ij,l+1} x_{ijk'}\right) + \sum_{i\in M} r_{ij,l+1} h_{ik'}, && l = 1, ..., m-1 \,(17) \\
& h_{in} + \sum_{j\in J} p_{ij} x_{ijk} \leq C_{max}, && \forall i \in M \,(18) \\
& h_{ik} \geq 0, && \forall i \in M, k = 1, ..., n \,(19) \\
& x_{ijk} \in \{0,1\}, && \forall j \in J, i \in M, k = 1, ..., n \,(20)
\end{aligned}
$$

Table 2.4: Rank-based model

The objective function is to minimize the C_{max}. The constraint (14) ensures that one job is assigned exactly to one position on each machine. The constraint (15) assumes that each job only gets one position on a machine. The constraint (16) implies that the start time of a job on a machine should be larger than the completion time of the previously scheduled job. The constraint (17) is the previously explained precedence constraint. The constraint (18) ensures that the makespan is at least the largest completion time of the last job on all machines. The constraint (19) claims that the start time of all jobs at all positions is greater or equal to 0. Many algorithms since the problem was described [35] follow the main constrains which implies the importance of the optimization in this area. All the models have in common the scheduling manufacturing approximation and the objective function that allow to compare them. It aims to demonstrate the diverse approaches modeling this kind of procedures.

In brief, the JSP is a difficult problem defined as an NP-hard problem that neither a nondeterministic Turing machine can solve in a polynomial time. It allows an increase in the number of solutions taking advantage of current technology such as parallel programming to improve industries [36]. Hence the motivation to introduce High-Performance Computing definitions. During this work HPC is the strategy to approximate to the best arrangement to get efficient in the manufacturing processes.

2.3 High-Performance Computing and Simulation

High-Performance Computing (HPC) uses the most robust computing procedures to solve the substantial technical and numerical problems that arise in simulations of complex physical systems, engineering applications, artificial intelligence training, big data, and multiple issues. The adaptive strategy of the HPC systems increases the recent applications in planning and scheduling [37]. Hence its importance in the Simulation of complex problems, especially with manufacturing simulation. In the following sections, the primary concepts of HPC are detailed to understand the HPC as a tool to approximate the best possible solution for complex problems.

2.3.1 HPC Theory

The HPC architecture determines how quickly the operations are performed. Nowadays, the different distributions and configurations of HPC increase the complexity of the systems. Thus, the HPC architecture is the organization and functionality of the components and the logical instructions that run programs on supercomputers. Three key properties determine the functionality of the HPC architecture: the parallelism of the system, the speed, and the efficiency of the components. The relation between those factors delivered in the performance of the HPC system is defined as follows:

$$P = e \times S \times a(R) \times \mu(E) \tag{2.8}$$

The formula 2.8 explains the relation between the main factors in the architecture of an HPC system, where the average performance is P, S is the scaling, a is the availability that depends on R that is the reliability, μ is the instruction retirement rate correlated with E that means a function of the power related to the consumption of energy.

Parallelism

The scalability achievement in High-Performance Computing systems is mainly based on its success in Parallelism [38] [39]. In simple words, Parallelism is the ability to do many things simultaneously. In HPC architecture, Parallelism is the capacity to perform multiple tasks simultaneously, thus reducing the total time to accomplish the operations. The control in Parallelism determines the performance in HPC architecture. Different frameworks improve this idea using new technologies such as the incorporation of GPUs to paralyze the process [40]. Thus, both the data and control paths are factors in how Parallelism improves the HPC architecture performance.

Efficiency

Another key in the performance of HPC is efficiency. Efficiency is the utilization of the system. Besides, it is defined as the percentage of time that critical components are employed. The common measure is the ratio of the sustained floating-point performance to the theoretical peak. The unit to measure is $flops$, floating-point operation per second.

The Equation 2.9 defines how to calculate the flops.

$$e_{flops} = \frac{P_{sustained}}{P_{peak}}; where\ \ 0 \leq e_{flops} \leq 1 \tag{2.9}$$

Speed

The third factor that influences performance is speed. Speed in this context refers to the speed of its components. For example, the speed of the processor determines the performance of the HPC. In this context, a key parameter is the clock rate that defines the rate at which each component retires instructions. Besides, bandwidth influences speed; it determines how much information moves between two components in a unit of time. The latency also relates to performance and how long the information moves between two points.

Reliability

Reliability normally is not considered a key factor. But, it determines how viable the system is. Errors may occur in two types. The first is *hard*, which implies when part of the system breaks permanently. And $soft$ when a failure occurs intermittently. The "noise" is one of the common causes of issues. It includes the number of hardware and software approximations such as low voltage margins or disconnection of nodes.

2.3.2 HPC Architecture

Figure 2.10 illustrates the different compositions of an HPC based on the Taxonomy of Flynx. They are distinct classes of parallel architectures. Flynn, in 1970, proposed a different class of parallel architecture, better known as the Taxonomy of Flynx [37]. The name refers to how many instructions and data streams the system can receive and process; D is a data stream, S is the stream, I is the instruction stream, and M is multiple data streams. The categories are the following: MIMD, multiple instruction streams, multiple data stream; MISD, multiple instruction streams, single data stream; SIMD, single instruction stream, multiple data stream; SISD, single instruction stream, single data stream.

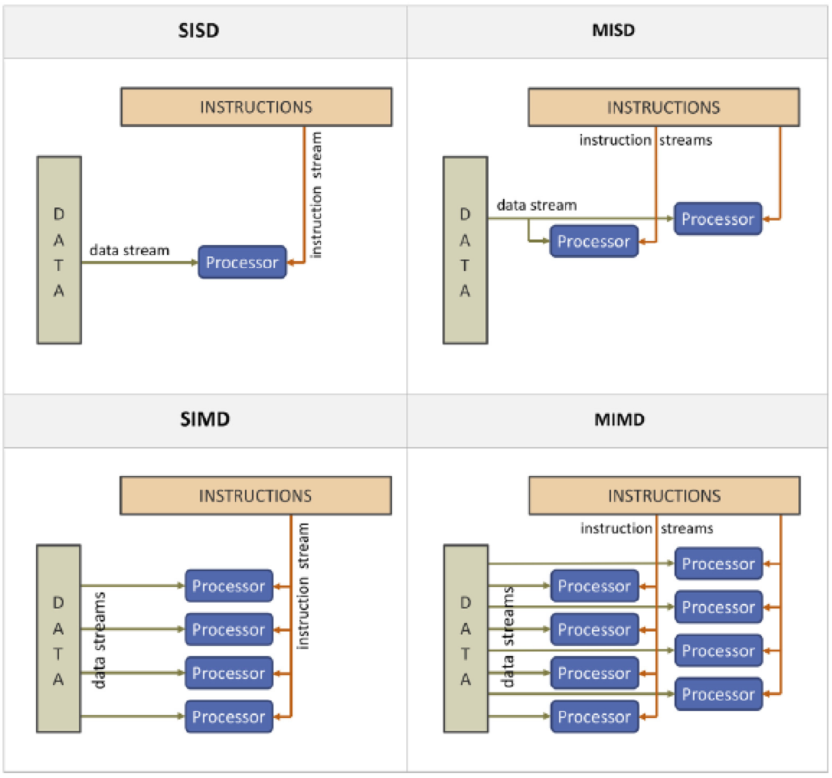


Figure 2.10: Taxonomy of Flynn: HPC Parallel Architectures. Source [3].

Figure 2.11 describes the main elements of an HPC. The distribution of different elements depends on each system. One the most common arrangement in HPC is known as *cluster*, it is relevant because it is the system that this work uses. A cluster is a set of equipment , called nodes, that execute the work simultaneously. Clusters aim to process problems with a great amount of data or complexity. The scalability of clusters [3] became an important part in their development, the easy incorporation of new processing nodes allowed to increase the systems and provide more computational power. The cluster characterizes possibly the single most successful form of supercomputing in the history of high-performance systems. It exploits technology progress in the areas of very large-scale integration microprocessors, dynamic random access memory (DRAM), and networking, as well as improving performance relative to cost through the economy of scale of mass production.

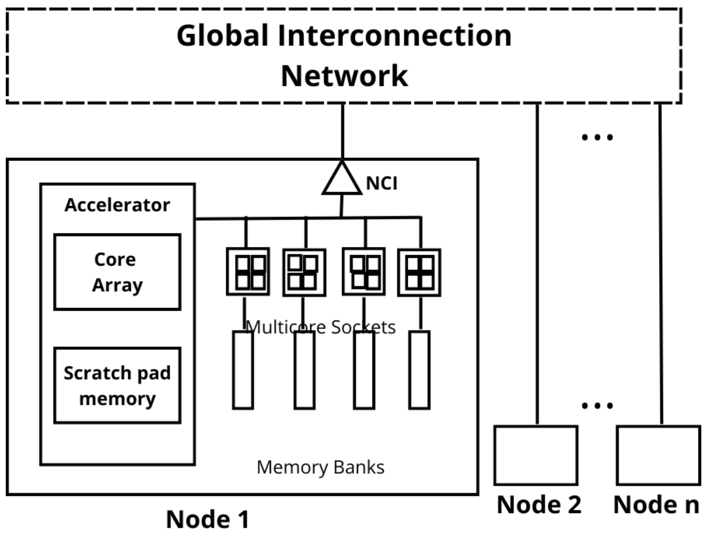


Figure 2.11: Elements in an HPC system.

Node

The node is a piece that incorporates all the functional elements required for computation. The node contains the major processing and main memory components; it includes a diversity of I/O controllers including but not limited to the system area network(s) (SANs). A node can perform independent user workloads enabling the scale increasing.

To enhance the acceleration architecture capabilities found in most ASICs and GPUs, a processor called Intelligence Processing Unit (IPU) was designed by Graphcore. IPU comes to be a fine-grained parallel processor designed to offer high performance for a wide range of computationally intensive ML algorithms. It is built with 1216 interconnected IPU tiles, which are interconnected through an 8 TB/s fabric on the chip, IPU-Links connect it, and it supports a maximum of 7296 parallel executions, which can be seen in Figure 2.12. The predecessor of IPU is a processor called Colossus GC2. It has IPU Tiles TM coupled In-Processor-Memory with an independent IPU-Core TM, IPU-Core TM. It supports 7296 programs running in parallel. Also, it has 45TB/s memory bandwidth, and the whole model is held on a chip, PCIe; it has 64GB/s bidirectional bandwidth to host, IPU-Exchange TM; non-blocking any communication pattern, IPU-Links TM; 300GB/s chip to chip bandwidth. It is a clear example of hardware improvement to achieve better simulation results.

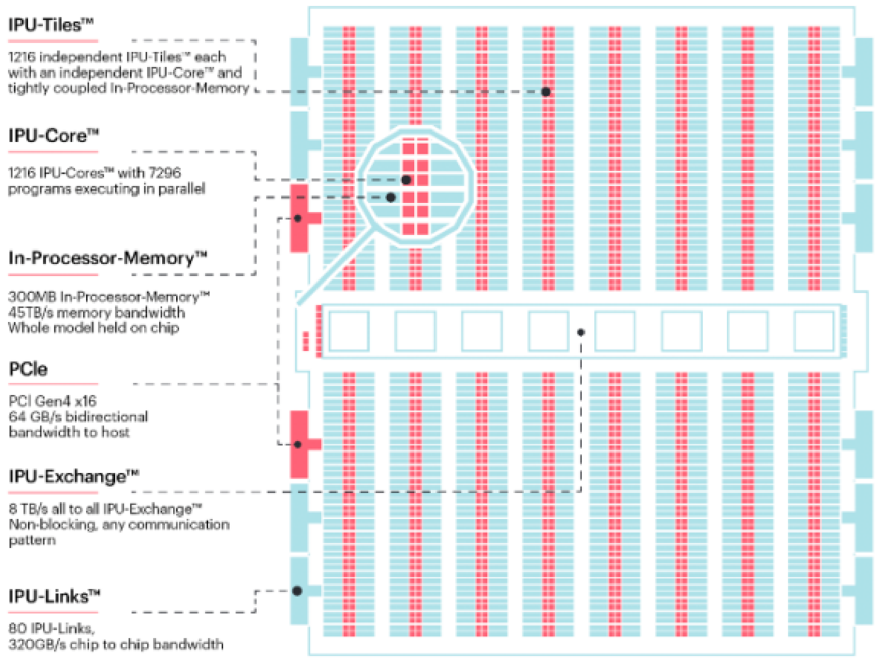

IPU-Tiles™
1216 independent IPU-Tiles™ each with an independent IPU-Core™ and tightly coupled In-Processor-Memory
IPU-Core™
1216 IPU-Cores™ with 7296 programs executing in parallel
In-Processor-Memory™
300MB In-Processor-Memory™ 45TB/s memory bandwidth Whole model held on chip
PCIe
PCI Gen4 x16 64 GB/s bidirectional bandwidth to host
IPU-Exchange™
8 TB/s all to all IPU-Exchange™ Non-blocking, any communication pattern
IPU-Links™
80 IPU-Links, 320GB/s chip to chip bandwidth

Figure 2.12: Intelligent Processing Unit (IPU) for the HPC configuration. Source [4]

2.3.3 HPC Applications

The latest generation commercial software is unable to complete a large scale dynamic simulation [41], due to the complexity of this model, but introducing parallelism in this process gives satisfactory results, since it shows great performance and accuracy, this was achieved with the help of eleven processors. This technique was tested in a WECC [42] planning base case with detailed models of synchronous generators. The techniques used were: parallelizing the calculation of generator current injection, identifying fast linear solvers for network solution, and parallelizing data outputs when interacting with APIs in the commercial package, TSAT. On the other hand, great tech companies also implement robust HPC systems as typical tools- One example is IBM, the IBM BlueGene/L [43] system is one of the high-end systems that is most likely to be subject to simulation. Besides, The simulations use hardware resources such as GPUs. GPUs help to make processes and simulations significantly better [44], but being very complex requires extensive knowledge to be able to use it. With this in mind, research was done with the Simian engine to facilitate the handling of GPUs, where effective results are shown because when using this engine the handlers go directly to the GPU, this happens thanks to the grouping and scheduling of the handlers.

Chapter 3

State of the Art

3.1 Simulation Approximation

Nowadays, new trends include the Internet of things, cloud computing, big data, virtual reality, artificial intelligence, industrial Internet of things, and blockchain. Thus, Smart Manufacturing Systems (SMSs) [45] have become the focus of attention to research for manufacturing optimization. Mr. Zhang [46] is one of many researchers that analyzed the influence of those technologies in the new era of the industry called industry 4.0. This new stage responds quickly to global competition and customization requirements. However, to get to this point, manufacturing has had different implementations over the years.

After a constant improvement over sixty years, the manufacturing industry has been incorporating elements that increase the complexity of its analysis. At the same time, the benefit of its results increases in parallel. The most recent trend in Manufacturing and Simulation is the incorporation of intelligent manufacturing systems [47] taking advantage of new technology features. The large influx of data generated by IoT systems opened a new path for integrating Big Data Analysis (BDA) [5] into manufacturing processes.Figure 3.1 shows in detail the workflow of a BDA functional model. It is important to emphasize that the tendency to visualize the results obtained from BDA is focused in a more graphical sense. Studies such as Zhuming's illustrate flexible and adaptive solutions for a guided environment primarily for analysis, diagnosis, and prevention within manufacturing processes. The holistic approach of Zhuming proves how BDA fits into Enterprise Architecture (EA). Today, Smart manufacturing systems use real-time data to improve decision accuracy, plant efficiency and performance, and overall productivity. However, processing the amount of data requires a system smart enough to control and predict based on the received information. On the other hand, another technology, such as Artificial Intelligence [48], is adapted in the manufacturing simulation environment. The AI approach shows good results in the use of Deep Learning [49] embedded in IoT. According to Li, using IoT-BDA and DL improves data prediction, making its prediction model 2.24 % more accurate than other models. It adapts CNN to improve the layers of predictions and obtain reliable results. CNN using parallelism is one novel way to tackle the BDA for manufacturing. It can handle enormous amounts of data in a reasonable period of time.

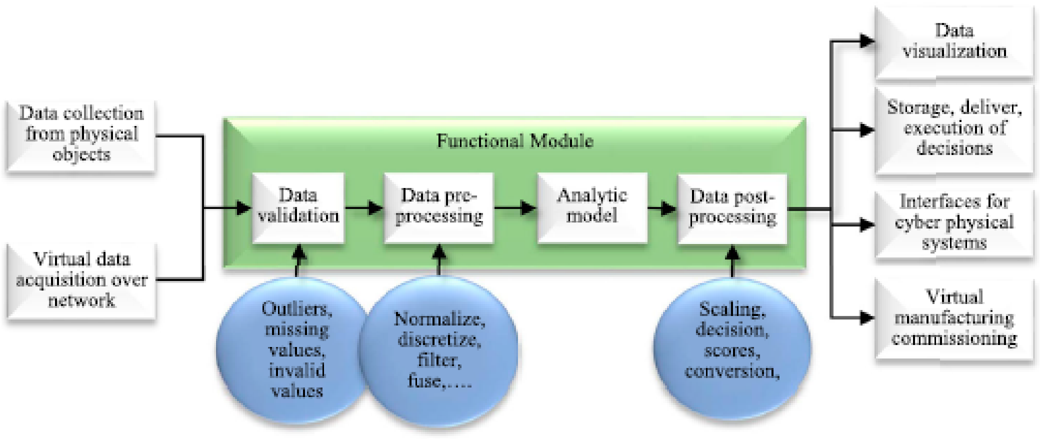

Data collection from physical objects
Virtual data acquisition over network
Functional Module
Data validation
Data pre-processing
Analytic model
Data post-processing
Outliers, missing values, invalid values
Normalize, discretize, filter, fuse,....
Scaling, decision, scores, conversion,
Data visualization
Storage, deliver, execution of decisions
Interfaces for cyber physical systems
Virtual manufacturing commissioning

Figure 3.1: Big Data Analysis workflow model for Digital Manufacturing. Source [5].

Another clear research trend is based on the creation of digital twins that allow, in conjunction with mathematical models, to predict the functioning of a physical environment. It is estimated that by 2025, the digital twins market will reach a record thirty-five trillion US dollars, and it is estimated that by 2027 it will double that figure. Furthermore, adding ML and AI [6] to the digital twins provides them with versatility for the manufacturing industry. According to Kosmas [6], these technologies provide intelligent analytics to these platforms facilitating labeling and robust data analysis autonomously. Figure 3.2 identifies different elements and modules that digital twin systems currently possess. The interconnection of modules and framework of this type of technology provides the opportunity to stand out from the current market trends.

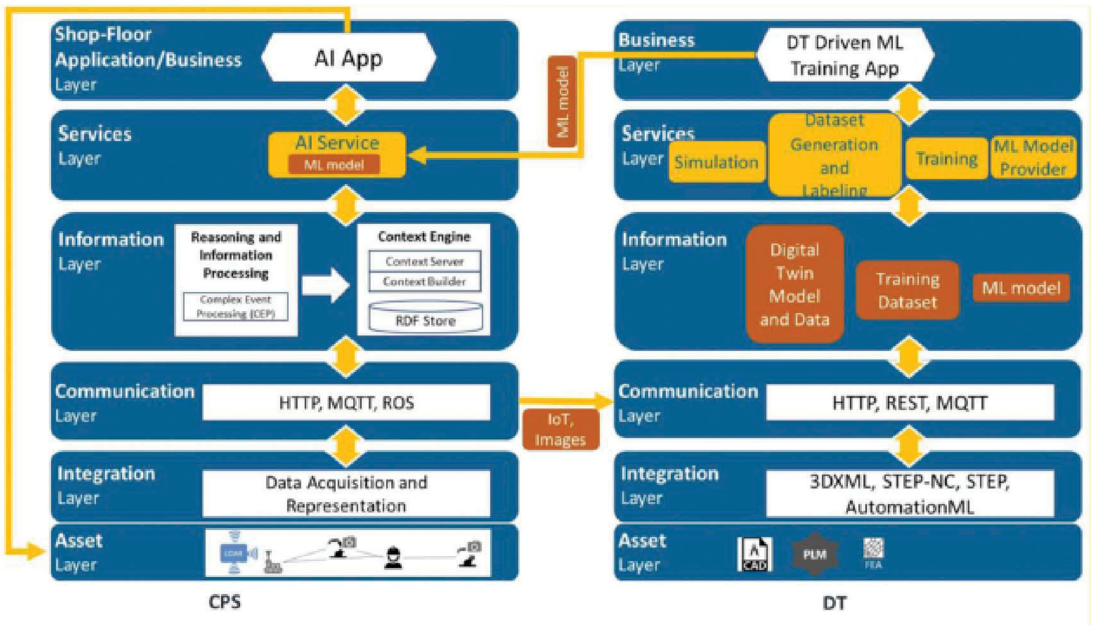


Figure 3.2: The digital twin for the development of ML-based applications for smart manufacturing. Source [6].

In the same way, Hasan and other researchers [7] present an alternative based on Ethereum technology to guarantee the following features decentralized, tamper-proof, immutable, and secure. In addition, technologies such as Blockchain [50] have also been incorporated into creating Digital Twins (DTs). Figure 3.3 demonstrates in detail an implementation for digital twins using blockchain. In addition, the use of digital contracts as a security mechanism is demonstrated. In recent years Blockchain has given new tools to the simulation such as smart contracts, security, autonomy, and record keeping. In short, Blockchain increases the trend of Digital Twin Networks (DTN), an edge related to the security of these digital environments that are the abstraction of a physical [51] world. Finally, blockchain in the Digital Twins section is not an isolated technology. Sahal [52] shows us that it is presented as an element for building collaborative frameworks that integrate various technologies that target use cases such as smart logistics and railway predictive maintenance.

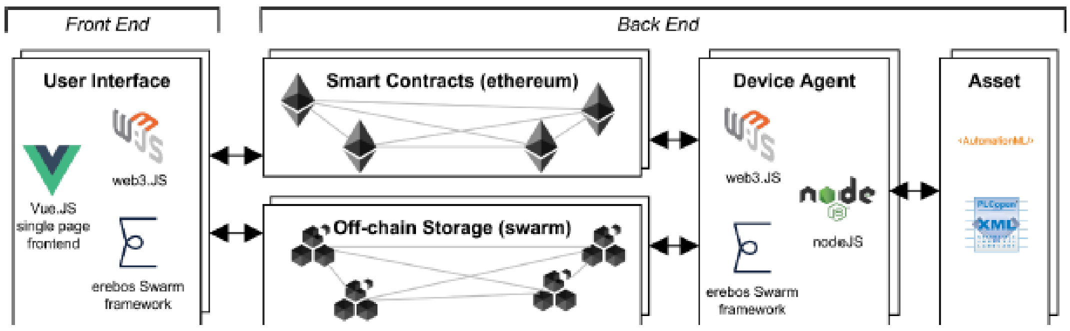


Figure 3.3: Implementation of Digital Twin using block-chain technology. Source [7].

Finally, in Discrete Event Simulation, the presence of open-source tools such as Salabim, JaamSim, and CloudSim [8] is the growing trend. Figure 3.4 shows the procedure using discrete event simulation on different platforms in a general scope. Lang [8] shows a use case compared with different tools with discrete event simulation as a core model, highlighting JaamSim as the best alternative due to its GUI implementation. The GUI section for modeling is an implementation that weighs on a commercial tool, given the ease of manufacturing engineering. Furthermore, this relates to programming skills since CloudSim is a purely code-based tool. On the other hand, tools such as Plant Simulation stand out for their genetic algorithm library, which is a plus in the case of optimization. Other tools, such as Arena, stand out for implementing the OptQuest engine, their competitive feature. The Table 1 attached in the the Appendix .1 describes features of tools based on discrete event simulation. Technologies such as Virtual Reality (VR) and 3D modeling stand out notably.

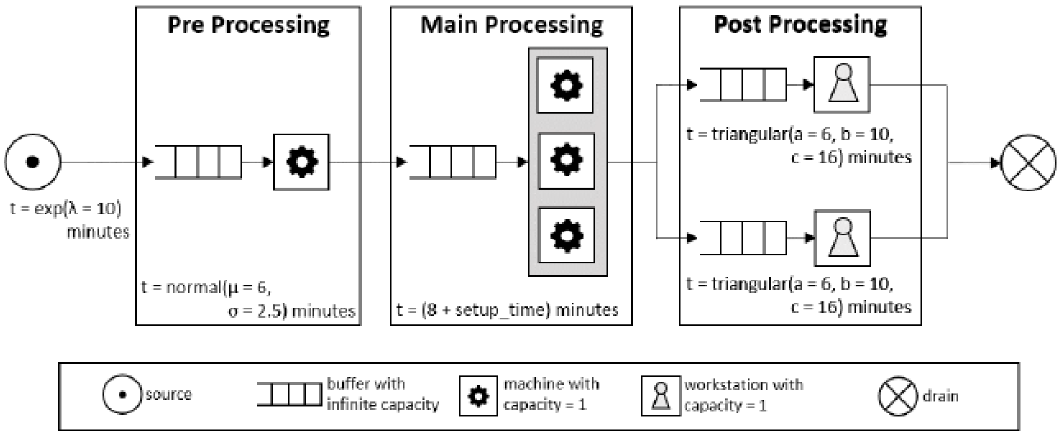

Pre Processing
Main Processing
Post Processing
t = exp(λ = 10) minutes
t = normal(μ = 6, σ = 2.5) minutes
t = (8 + setup_time) minutes
t = triangular(a = 6, b = 10, c = 16) minutes
t = triangular(a = 6, b = 10, c = 16) minutes
source
buffer with infinite capacity
machine with capacity = 1
workstation with capacity = 1
drain

Figure 3.4: Conceptual model of the reference problem case. Source [8]

3.2 HPC Strategy

The field of High-Performance Computing is constantly evolving new technological features maximizing the use of resources. The improvements in the HPC area lies in taking advantage of hardware and computational power to find solutions for complex problems. Hence, the connection with Simulation that implies the digital abstraction of the physical world, increasing the demand of computational resources. Simulation in HPC systems open a new pathway of research and application to the industry.

Mainly introduced to graphics renderization, GPUs now are general-purpose hardware to speed up computational processes. One example of such a model is the work of Faheem *et al.* [53], introducing the use of Graphical Processing Units (GPUs) to accelerate the application in the simulation. Based on Discrete Event Simulation (DES) code in python, the work proposes a comparative approach between DES and GPUs. The researchers use the reinforcement learning strategy in the models; Business Process Modeling and Notion (BPMN) and Linear Balancing Model. They compare three groups with small, medium, and large data to simulate. The results show that the TensorFlow-GPU and TensorFlow-CPU have similar performance comparing results; however, the GPU approximation took less time working with small and medium computation sizes. An anomaly is described in this work when a complex line-balancing model conveyed that TensorFlow-CPU performed better than TensorFlow-GPU. They suggest deepening this anomaly to find a better initial tune of parameters.

One of the trends that will be used to have a better response in numerical simulation and artificial intelligence workflows is general purpose processors (GPP) [4]. The combination of accelerators and GPPs illustrates future trends that can be expected in the coming years. Another trend is to improve CPU-GPU interconnection by using some Intel products with BFLOAT16 support and the acquisition of ARM by NVIDIA. The design of a CPU-accelerated node without needing a DDR is proposed. Thus, the HBM (High Bandwidth Memory) of the GPU will have direct access to the consumed memory. The interconnection inside the node will allow a smoother and closer integration between CPU and GPU, minimizing the data movements between them; it guarantees fluidity in the cache memory.

Other authors use HPC systems with other ideas, such as the case of Kumbhare *et* *al.* [54]. Their study introduces the hybrid-VPT, a new power allocation strategy. It is a value-based heuristic in a power-constrained HPC environment. They compare different models using a percentage of all 2048 nodes to identify the more adaptive power allocation strategy. Under the tightest power constraint of 55%, VPT-JSPC and hybrid-VPT have the most favorable improvement using HPC. Otherwise, when the power constraint is 70%, VPT-CPC and hybrid-VPT are more favorable than VPT-JSPC. This work demonstrates the value-oriented approach for Job Shop Scheduling for power constraint HPC systems. In brief, the hybrid-VPT model demonstrates a better performance under limiting conditions of the system.

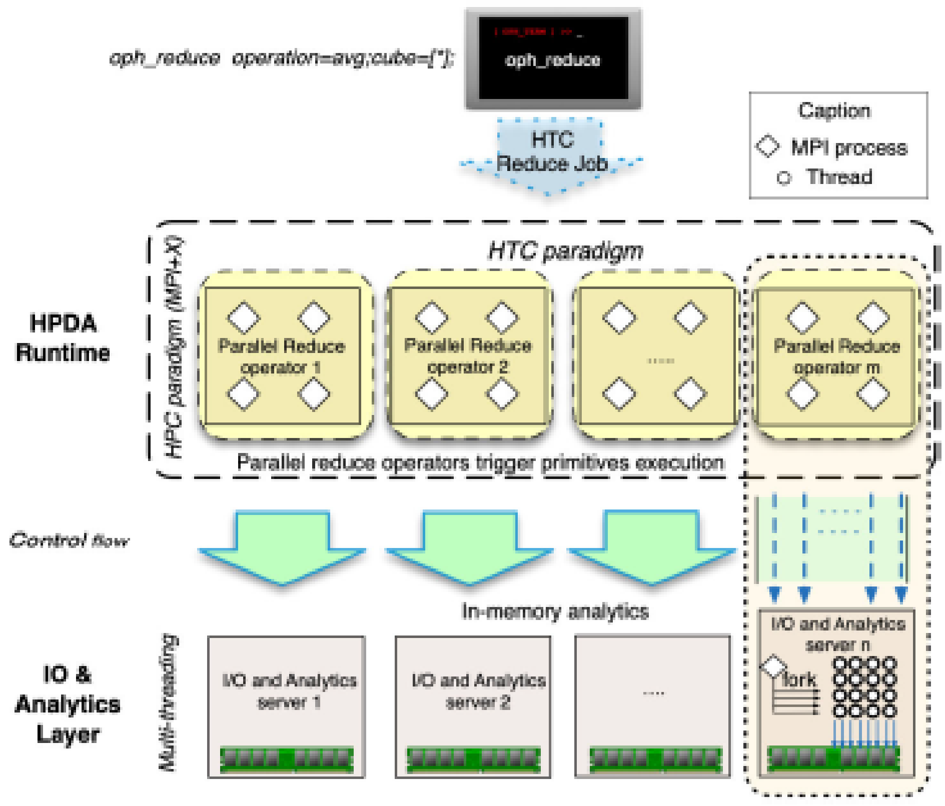

oph_reduce operation=avg;cube=[*];
oph_reduce
HTC Reduce Job
Caption
MPI process
Thread
HTC paradigm
HPDA Runtime
HPC paradigm (MPI+X)
Parallel Reduce operator 1
Parallel Reduce operator 2
Parallel Reduce operator m
Parallel reduce operators trigger primitives execution
Control flow
In-memory analytics
IO & Analytics Layer
Multi-threading
I/O and Analytics server 1
I/O and Analytics server 2
I/O and Analytics server n
fork

Figure 3.5: High-level representation of the proposed HPDA runtime system for a data reduction job. Source [9].

On the other hand, another application of HPC in simulation is using High-Performance Data Analytics (HPDA) [9] to perform forecasting simulations. A clear example is the research of D. Elia in Europe, where he uses HPDA to improve the system execution, optimization in task execution, and the optimization in the data predictions. He implemented

Artificial Intelligence (AI) to address intelligence-driven data-centric scenarios. His team researched the use of HPC technology and HPDA to offer HPDAaaS as a service. Figure 3.5 shows the high-level representation of the proposed HPDA execution system for a data reduction job. It also shows the parallel work performed by the memory in order to reduce the complexity of the problem. The HPC is the software ecosystem or environment, while the HPDA is the data-centric paradigm using Big Data as an application in supercomputing infrastructures [55]. HPDA is coupled with the Ophidia framework to provide an enabling environment for Big Data. In other words, It shows multi-dimensional data modeling, parallel and in-memory data processing, data-driven task scheduling, and a service-oriented interfaces system.

Chapter 4

Methodology

4.1 Overview of the Proposed Solution

4.1.1 Description of the Problem

The problem in simulating manufacturing processes lies in achieving a reliable computational abstraction of the production model. Optimization and automation became key points to achieve even ambitious goals such as the industry 4.0 [11] hence the importance of seeking technological approaches to obtain results. The complexity of the problem increases because of the flexibility in the production process, especially in the scheduling area. The manufacturing industry commonly uses a production sequence that can be modeled as a Job Shop Problem JSP). The JSP model implies different machines doing specific tasks in a manufacturing sequence to obtain a final result (See Section 2). A great applicable example is in the oil manufacturing process, the machines follow a specific order of instructions to obtain one final product, gasoline, guided by previous specifications. The goal is to optimize the resources during the production, knowing the best time arrangement in each step. However, the problem, the JSP, is a very complex one with a computational approach (See Section 2.2.2). Hence, the complexity resembles the Job shop scheduling problem (JSP), which has been proved to be an NP-Hard problem for more than three tasks in two machines. In simple words, the complexity of the problem is exponential due to the number of variables that must be considered [29].

4.1.2 Solution Background

Due to the complexity of the proposed problem, there are several ways to approach or propose a solution. This work will focus on the approach based on disruptive technological tools. This approach proposes a variant to contribute to the search for a solution to the problem based on the use of hardware and software resources. In this work, a solution based on discrete event simulation indicates a considerable reduction in the complexity of the problem. It is because, unlike a continuous simulation, the discrete simulation decreases the computational resources needed in complex environments. Continuous simulation models the system as a sequence of events that occur in time; thus, continuous follows all the procedures over time. Then, continuous simulation is for simple continuous modeling, which limits its complex implementation. In contrast, DES has the advantage of modeling complex systems with variable stochastic events due to the scalability over time. Thus, it can support simultaneous processing sequences, such as industrial processes. Finally, It is important to emphasize that technology, such as high-performance computing, is also considered. This approach compares a typical implementation with an implementation that can use more computational resources. The technologies mentioned above align with the growing trend of Industry 4.0, which seeks to solve complex problems through advanced technological tools. Something to keep in mind is that the solution proposed below is a method for less automated industries that seek to improve the optimization of their resources and performance. To sum up, it is essential to emphasize that the elements realized in this work are highly replicable for companies with low automation and technological resources.

4.2 Methodology Design

The methodology carried out during this work defines the project in three main phases. Figure 4.1 illustrates the three main plashes and the sequential procedure. The first one is the Environment Settings phase. During this phase, in order to understand the simulation method, the following explanation related to the production process is a necessary step. This method bases its functionality on modeling the industry processing using programming tools and mathematical modeling. Thus, section 4.2.1 describes the manufacturing process. Therefore, the section 4.2.2 explains the artificial environment operation. The second phase is the Serial Implementation, during this step the code implementation is defined. It refers to program the discrete event simulation engine. To sum up, the second phase is the basic implementation to check the model accuracy. The section 4.2.3 explains the serial implementation and expected results during this phase. Finally, the third phase implies to implement the code in its parallel version. To analyze the results and identify the best optimization of resources to perform the simulation. The section 4.2.5 identifies the key points in the parallel implementation using the HPC system.

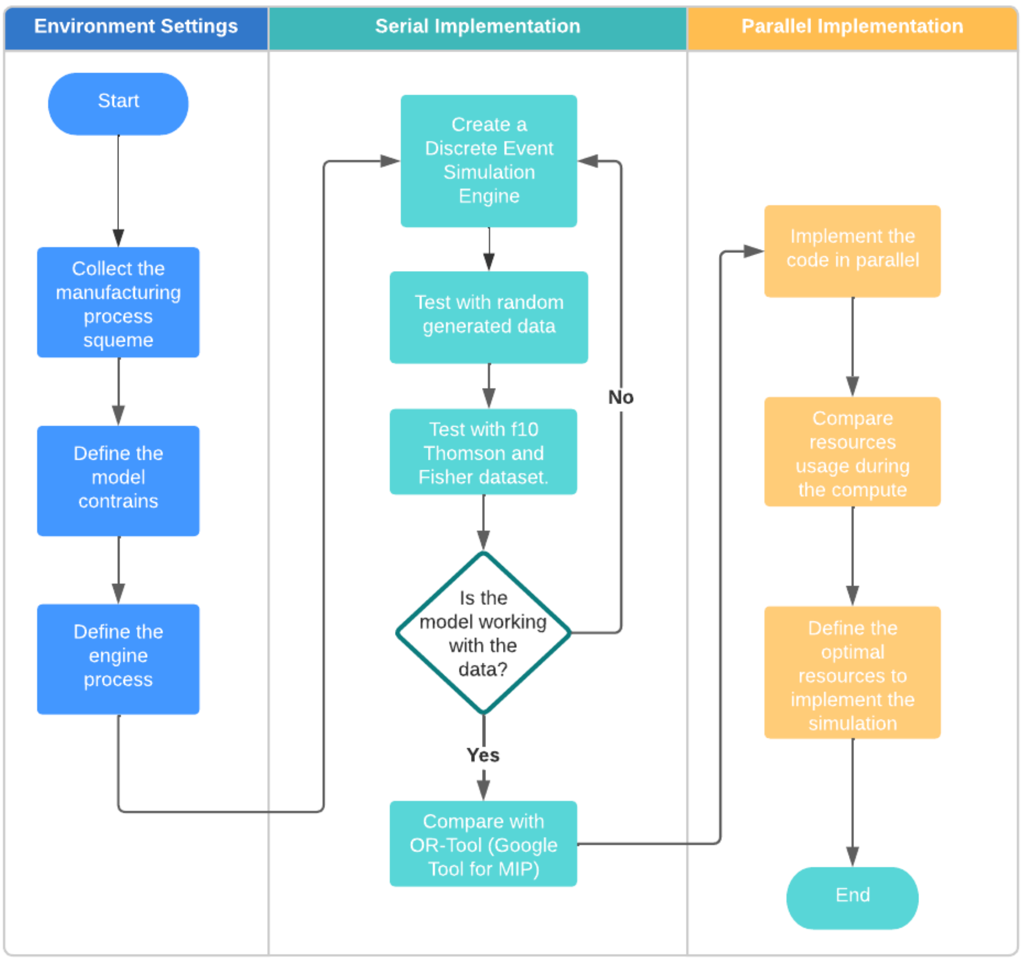

Environment Settings
Serial Implementation
Parallel Implementation
Start
Collect the manufacturing process squeme
Define the model contrains
Define the engine process
Create a Discrete Event Simulation Engine
Test with random generated data
Test with f10 Thomson and Fisher dataset.
Is the model working with the data?
No
Yes
Compare with OR-Tool (Google Tool for MIP)
Implement the code in parallel
Compare resources usage during the compute
Define the optimal resources to implement the simulation
End

Figure 4.1: Three phases of the Methodology Design.

4.2.1 Production Process

The production process is a multistage procedure. It contains the coordination of the inputs using a scheduling process. The inputs for this method are the raw material, the production order, and the production line process. In brief, finding the best time arrangement includes knowing the expected results, the procedure, and the material. Figure 4.2 shows the joint action plan for a manufacturing industry. The output of this sequence is the proper schedule to obtain the products by optimizing the resources.

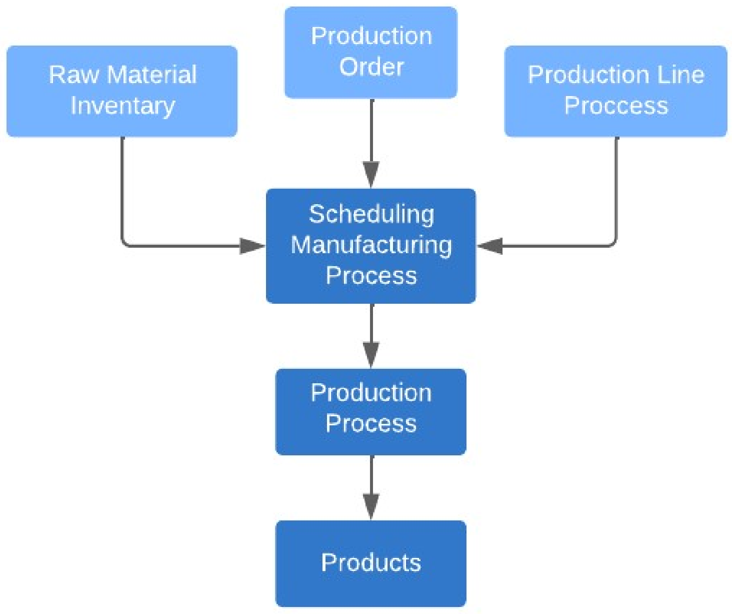


Figure 4.2: Main Steps in the Manufacturing Process in a common Industrial Implementation.

4.2.2 Simulation Engine

Figure 4.3 denotes the core of the simulation steps. First, it begins reading the input data, the initialization data, or the variables settings. Then the process tests if there is any next activity. The method sets the activity with the new activity if there exists. Otherwise, it will proceed to the end. Then, the activity needs an available machine to perform the task. Therefore, the function gets a machine that seeks any available machine that fits the activity characteristics. After the action finds a machine, it sets the initial time to identify the time spent during the process. Thus, the machine runs the simulation. Finally, at the end of the activities, the method collects the simulation environment and creates the necessary reports.

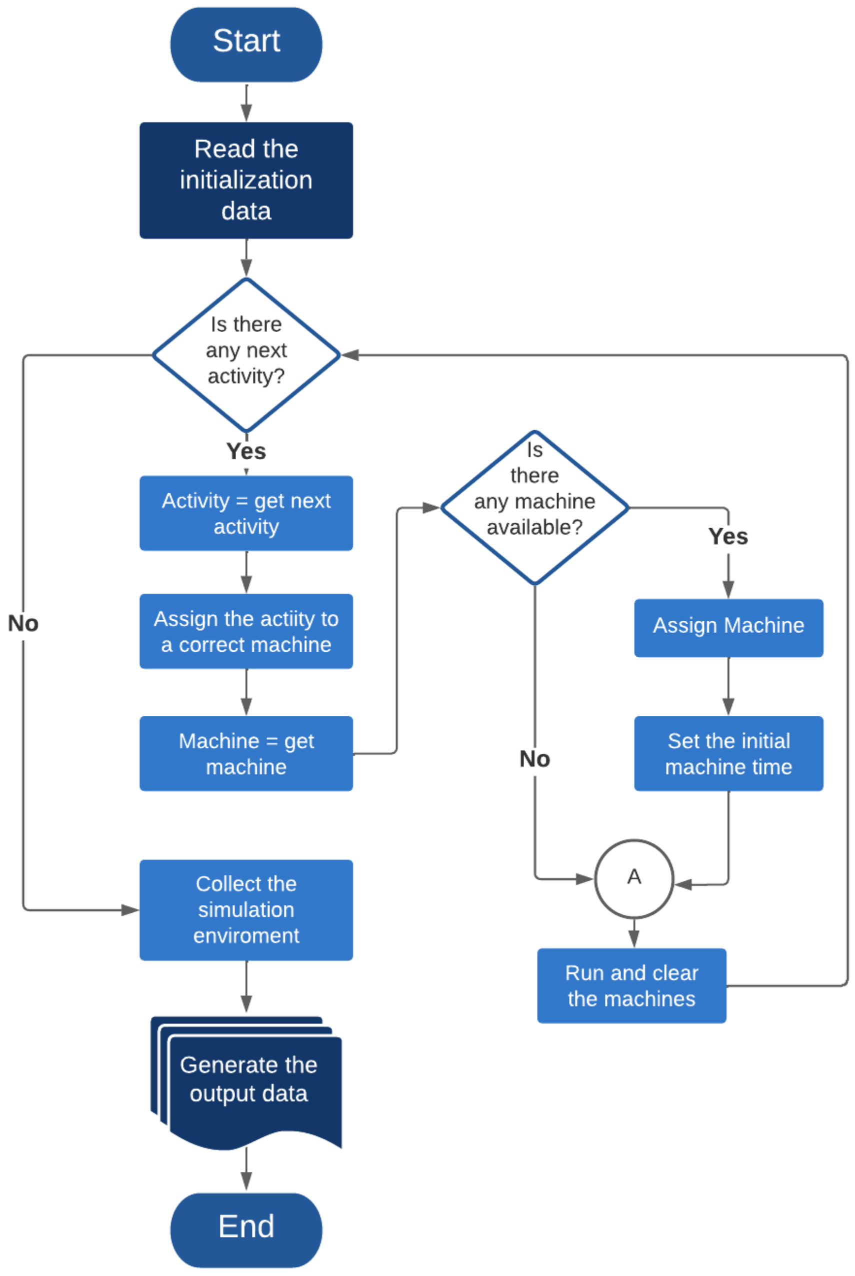

Start
Read the initialization data
Is there any next activity?
Yes
No
Activity = get next activity
Assign the actiity to a correct machine
Machine = get machine
Is there any machine available?
Yes
No
Assign Machine
Set the initial machine time
A
Run and clear the machines
Collect the simulation enviroment
Generate the output data
End

Figure 4.3: Simulation Engine Main Steps.

4.2.3 Discrete Event Simulation Serial Implementation

Section 4.2 denotes the design of the current methodology. During the first phase, the environment and constraints are defined. During the second phase, the DES serial imple-

```
Data: $ft06 - ft10 - ft20$ Dataset
Result: Scheduled Activities

Class jobs;
    $id \leftarrow$ job id;
    $properties \leftarrow$ job properties;
    $job\_completed \leftarrow False$;
    $step \leftarrow 0$;
    $release\_time \leftarrow []$;
Class machine;
    $id \leftarrow machineid$;
    $busy \leftarrow availability$;
    $rtime \leftarrow$ release time $= 0$;
    $properties \leftarrow []$;
    $job\_task\_attempted \leftarrow []$;

$job\_sequence = []$;
$machine\_activity = []$;
$jobs = listofjobs$;
$machines = listofmachines$;
$event\_timer = 0$;
$sort\_jobs\_by\_length(ascending)$;

while $jobs \leq jobs\_count$ do
    $sort\_machines\_by\_earliest\_release$;
    $check\_idle\_machines$;
    if $machines[0].release\_time < event\_time$ then
        $free\_machine()$;
        if machine[0].availability == True then
            $machine[0] \leftarrow job[i]$
        else
            $free\_machine()$
        end
    else
        if job is assigned then
            $jobs[i].waiting\_time = event\_timer - job[i].arrivaltime$;
            $jobs[i].complete\_time = event\_timer + job[i].duration$;
        end
        $job\_sequence.append(jobs[i])$;
        $machine\_activity.append(machine[0])$;
    end
    $i = i + 1$
end
```

Algorithm 1: Pseudo code for the serial implementation of the Discrete Event Simulation

mentation considers the ft06, ft10, and ft20 data sets (See Section 4.3.1 for data set details). Furthermore, Figure 4.3 demonstrates the engine architecture for the DES serial implementation. Finally, the computational logic follows the discrete simulation steps based on the Disjunctive Model previously explained in Chapter 2. In order to be able to recreate the simulation, Python is the tool selected to be able to abstract the main characteristics of the environment to be simulated. In addition, Python allows the creation of a simulation and modeling tool to support manufacturing industry characteristics. Finally, Discrete Event Simulation will have two commissions. The first will be the serial, and the second will be the parallel procedure. Since then, to clarify the terminology, Discrete Event serial and parallel simulation will be serial and parallel implementation (DES serial and DES parallel).

Algorithm 1 shows the main steps for the Discrete Event Simulation (DES) implementation. It remarks on the steps for the serial Discrete Event Simulation. The implementation defines two classes, the first one for the job characteristics and the second one for machine characteristics. The first class contains the id of the job to track the optimization and the job requirements in the properties item. Then the class initializes the step and the completed. The second class defines the machine and its characteristics. Each machine has one id to track the process with the job and machine id. The class also checks the availability and sets the release time. Besides, properties and the job task attempted are declared. The next step is to create the list to save the simulation results. The code also sets the unit of time as a critical characteristic for DES. Finally, the simulation process is performed based on the simulation engine and model restrictions. The output of the process is the best possible arrangement of times of the jobs in the machines.

4.2.4 OR-Tool Serial Implementation

The methodology also considers another optimization tool to compare the results from the local implementation and the tool. OR-Tools is an open-source software suite for optimization, tuned for tackling the globe's most demanding problems in vehicle routing, flows, integer and linear programming, and constraint programming. It is a Google tool powered by Google AI to solve the most computational demanding linear programming, mixed integer programming, and constraint programming. It is important to remark that

OR-Tools has won gold in the international constraint programming competition every year since 2013, it is consider the best optimization tool [56]. The OR-Tools serial implementation executes a compare method with the design Discrete Event Simulation. Then, The Algorithm 2 shows the functions to implement the tool for the ft06, ft10, and ft20 data sets. OR-Tools is the best optimization tool. Thus, comparing the DES method with the OR-Tools will detail critical points of this approach. The model uses Constraint optimization and the CP-SAT Solver. OR-Tools provides the MPSolver wrapper for solving scheduling problems.

```
Data: $ft06 - ft10 - ft20$ Dataset
Result: Scheduled Activities
Import ortools.sat.python;
  $machines\_count \leftarrow 1 + max\_jobs$;   $all\_machines \leftarrow machines\ count$;
 $horizzon \leftarrow sum\ task\ in\ jobs\_data$; $model \leftarrow cp\_model.CpModel$;
   $task\_type \leftarrow collections.namedtuple$;
   $assigned\_task\_type \leftarrow assigned\_task\_type,\ startjob\_indexduration$;
$all\_task \leftarrow []$; $machine\_intervals \leftarrow collections.defaultdict$;
for job_id in enumerate(jobs_data) do
    for task in enumerate(job) do
        $machine \leftarrow task[0]$; $duration \leftarrow task[1]$;
        $start\_var \leftarrow model.NewIntVar - start$;
         $end\_var \leftarrow model.NewIntVar - end$; $all\_tasks \leftarrow task\_type$;
         $machine\_to\_intervals \leftarrow interval_var$;
    end
end
for machine in all machines do
    model.AddNoOverLap(machine_to_intervals[machine])
end
for job_id, job in enumerate(jobs_data) do
    for task_id in range(len(job)-1) do
        model.Add(all_tasks[job_id, task_id + 1].start $\geq$ all_tasks[job_id,
         task_id].end)
    end
end
$obj\_var \leftarrow model.NewIntVar$;
model.AddMaxEquality(obj_var,[ all_tasks[job_id, Len(job) - 1].end]);
for jobid, job in enumerate(jobs_data) do
    model.Minimize(objvar)
end
$solver \leftarrow cp\_model.CpSolver()$; $status \leftarrow solver.Solve(model)$;
```

Algorithm 2: Pseudo code with the Google OR-Tool implementation.

4.2.5 Parallel Implementation

After developing the serial implementation, the third phase compares the serial and parallel versions. For this section, Algorithm 3 explains the routine to obtain the computed metrics. The system uses a cluster which is considered a High-Performance Compute system. The parallel version uses GPUs to provide more resources and create an environment to compare the computational resources used. For this implementation, the OpenCL tool [57] creates the Kernel to use in parallel implementation the GPU using the same computational logic as the Algorithm 1.

```
Data: none
Result: $Compute\ Statistics$
Import $pyopencl\ pynvml\ psutil$;
Function $pynvml.nvmlInit$;
    $device\_count \leftarrow pynvml.nvmlDeviceGetCount()$;
    $handle \leftarrow pynvml.nvmlDeviceGetHandleByIndex(0)$;
 end $interval \leftarrow 0$;
$gpu\_usage \leftarrow []$; $gpu\_threads \leftarrow []$; $gpu\_memory \leftarrow []$; $cpu\_usage \leftarrow []$;
$cpu\_threads \leftarrow []$; $ram\_usage \leftarrow []$; $ram\_percent \leftarrow []$;
Function $log\_usage$;
    $current\_time \leftarrow time$;
    $mem\_info \leftarrow pynvml.nvmlDeviceGetMemoryInfo$;
    $gpu\_usage \leftarrow mem\_info.used$;
    $gpu\_memory \leftarrow utilization_rates.gpu$;
    $cpu\_usage \leftarrow psutil.cpu_percent$;
    $cpu\_threads \leftarrow psutil.cpu_count$;
    $cpu\_threads \leftarrow psutil.cpu_count$;
    $ram\_info \leftarrow psutil.virtual\_memory$;
    $ram\_usage \leftarrow ram\_info.used$;
    $ram\_percent \leftarrow ram\_info.percent$;
end
```

Algorithm 3: Pseudo Code for the Monitor metrics functions.

4.3 Experimental Setup

4.3.1 Data set

The data sets ft06, ft10, and ft20 represent a prevalent benchmark job shop industrial problem. Thus, the data set was exposed in the article ”Probabilistic learning combinations of local job-shop scheduling rules” by H. Fisher [58]. In addition, the well-known data set has been used as a benchmark to optimize the makespan function in different papers, and research works [59] [60], showing the efficiency of the data.

Job	**T0**	**T1**	**T2**	**T3**	**T4**	**T5**
job0	(2, 1)	(0, 3)	(1, 6)	(3, 7)	(5, 3)	(4, 6)
job1	(1, 8)	(2, 5)	(4, 10)	(5, 10)	(0, 10)	(3, 4)
job2	(2, 5)	(3, 4)	(5, 8)	(0, 9)	(1, 1)	(4, 7)
job3	(1, 5)	(0, 5)	(2, 5)	(3, 3)	(4, 8)	(5, 9)
job4	(2, 9)	(1, 3)	(4, 5)	(5, 4)	(0, 3)	(3, 1)
job5	(1, 3)	(3, 3)	(5, 9)	(0, 10)	(4, 4)	(2, 1)

Table 4.1: ft06 data set from Fisher and Thompson

Table 4.1 shows the ft06 data set. The first column represents the jobs and their respective IDs. The first row represents the task ID. Each value in the table indicates the individual steps of each job. Every value has two numbers, the first one represents the number of steps to procedure in the machine while the second value represents the amount of time units. In brief, ft06 data set are 6 machines, 6 jobs with 6 steps each.

Job	**T0**	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**
job0	(0,29)	(1,78)	(2, 9)	(3,36)	(4,49)	(5,11)	(6,62)	(7,56)	(8,44)	(9,21)
job1	(0,43)	(2,90)	(4,75)	(9,11)	(3,69)	(1,28)	(6,46)	(5,46)	(7,72)	(8,30)
job2	(1,91)	(0,85)	(3,39)	(2,74)	(8,90)	(5,10)	(7,12)	(6,89)	(9,45)	(4,33)
job3	(1,81)	(2,95)	(0,71)	(4,99)	(6, 9)	(8,52)	(7,85)	(3,98)	(9,22)	(5,43)
job4	(2,14)	(0, 6)	(1,22)	(5,61)	(3,26)	(4,69)	(8,21)	(7,49)	(9,72)	(6,53)
job5	(2,84)	(1, 2)	(5,52)	(3,95)	(8,48)	(9,72)	(0,47)	(6,65)	(4, 6)	(7,25)
job6	(1,46)	(0,37)	(3,61)	(2,13)	(6,32)	(5,21)	(9,32)	(8,89)	(7,30)	(4,55)
job7	(2,31)	(0,86)	(1,46)	(5,74)	(4,32)	(6,88)	(8,19)	(9,48)	(7,36)	(3,79)
job8	(0,76)	(1,69)	(3,76)	(5,51)	(2,85)	(9,11)	(6,40)	(7,89)	(4,26)	(8,74)
job9	(1,85)	(0,13)	(2,61)	(6, 7)	(8,64)	(9,76)	(5,47)	(3,52)	(4,90)	(7,45)

Table 4.2: ft10 data set from Fisher and Thompson

Table 4.2 shows the Fisher & Thompson FT10 data set, which comprise ten machines, ten jobs, and ten steps each. Each of the data shows the job that the machine performs, represented by the first column of the table, while the first row of the table indicates the time that this job takes in each process performed by the machine. In each cell, the Job value is represented on the left and the time value on the right. Besides, the optimal optimization for makespan objective function is 930.

Job	T0	T1	T2	T3	T4
job0	(0, 29)	(1, 9)	(2, 49)	(3, 62)	(4, 44)
job1	(0, 43)	(1, 75)	(3, 69)	(2, 46)	(4, 72)
job2	(1, 91)	(0, 39)	(2, 90)	(4, 12)	(3, 45)
job3	(1, 81)	(0, 71)	(4, 9)	(2, 85)	(3, 22)
job4	(2, 14)	(1, 22)	(0, 26)	(3, 21)	(4, 72)
job5	(2, 84)	(1, 52)	(4, 48)	(0, 47)	(3, 6)
job6	(1, 46)	(0, 61)	(2, 32)	(3, 32)	(4, 30)
job7	(2, 31)	(1, 46)	(0, 32)	(3, 19)	(4, 36)
job8	(0, 76)	(3, 76)	(2, 85)	(1, 40)	(4, 26)
job9	(1, 85)	(2, 61)	(0, 64)	(3, 47)	(4, 90)
job10	(1, 78)	(3, 36)	(0, 11)	(4, 56)	(2, 21)
job11	(2, 90)	(0, 11)	(1, 28)	(3, 46)	(4, 30)
job12	(0, 85)	(2, 74)	(1, 10)	(3, 89)	(4, 33)
job13	(2, 95)	(0, 99)	(1, 52)	(3, 98)	(4, 43)
job14	(0, 6)	(1, 61)	(4, 69)	(2, 49)	(3, 53)
job15	(1, 2)	(0, 95)	(3, 72)	(4, 65)	(2, 25)
job16	(0, 37)	(2, 13)	(1, 21)	(3, 89)	(4, 55)
job17	(0, 86)	(1, 74)	(4, 88)	(2, 48)	(3, 79)
job18	(1, 69)	(2, 51)	(0, 11)	(3, 89)	(4, 74)
job19	(0, 13)	(1, 7)	(2, 76)	(3, 52)	(4, 45)

Table 4.3: ft20 data set from Fisher and Thompson

Table 4.3, in the same way as the previous tables, shows the Job performed by the machine together with time; these, in turn, are represented by the first column and the first row of the table, respectively. The cells contain the values of job and time. This specific table has 20 Jobs and 5 Tasks each. The ft20 data set is the most challenging in computational concepts. Due to the arrange of data needed to optimize this resources.

4.3.2 Serial Experimental Setup

The procedure for this section is done on a LAPTOP-IEBFJKB9\ LENOVO, with Microsoft Windows 11 Home, Version 10.0.22000 Build 22000, the system type is x64-based PC, the processor is 11th Gen Intel(R) Core(TM) i3-1115G4 @ 3.00GHz, 2995 Mhz, 2 Core(s), 4 Logical Processor(s), in addition to having Random Access Memory of 8.00 GB

(RAM).

4.3.3 Parallel Experimental Setup

The procedure for this section uses a cluster with enough computational characteristics to be considered as High Performance Computing system. To carry out this project, the computer used met the following characteristics, CPU op-modes of 32-bit, 64-bit, with an Architecture of x86_64, with 4 CPU(s), the Model name is Intel(R) Xeon(R), Model: 79, CPU MHz: 2199.998 and CPU @ 2.20GHz. Besides, the GPU charateristics are GPU Tesla P100-PCIE-16GB - NVIDIA CUDA with 16GB of GPU Memory and the overall performance of 4.1 TFLOPS.

Chapter 5

Results

In this section, the figures and tables represent the schedule-optimized structures based on simulation using the benchmark data sets ft06, ft10, and ft20. The scheduling issue aims to obtain the optimal solution to reduce cost and raw materials. The Gantt chart will mainly explain the workflow distribution and explains the machine and job distributions all in one chart. The chart seeks to visualize the number of jobs and times to plan future industrial activities based on computer simulation. The second phase of the methodology generates a serial simulation to compare the best optimization tool from Google called OR-Tool and the Discrete Event Simulation low-cost implementated in this work. After this procedure, the results of the ft20 data set implementation are parallelized to analyze the benefits and limitations of having one HPC system. Finally, to clarify during the results the time consider in each figure is in Units of Time (ut).

5.1 OR-Tool

This section describes the results of the OR-Tool implementation using the benchmark data sets. Each data set has one table with the results specifications and one Gantt Chart to visualize the same results. The label for each job is described in the following format job0_task_1, describing the specific item for each machine. The tables describe from left to right the workflow in each machine. Therefore, the two numbers represent the start unit of time and the second the end unit of time.

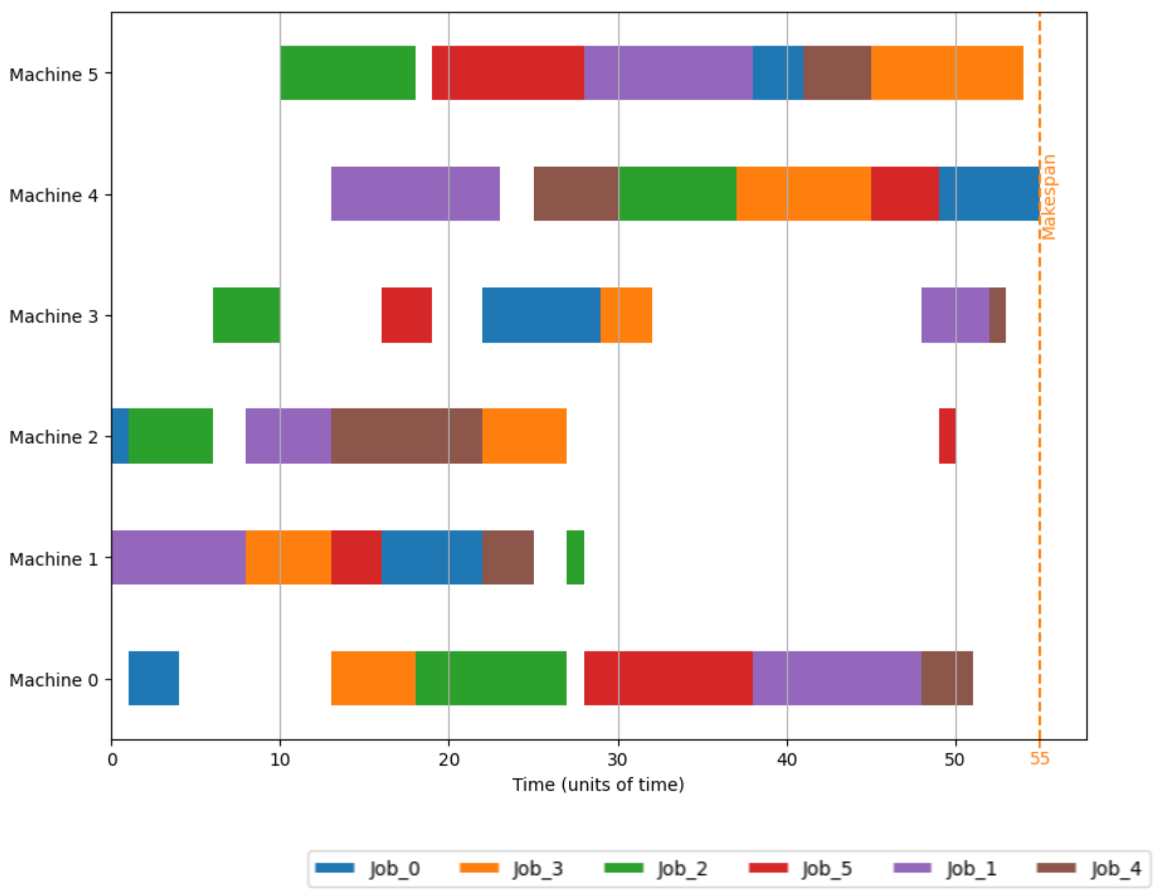


Figure 5.1: Serial Simulation with Google OR-Tool. ft06 Thomson and Fisher Data-set.

Table 5.1 shows the results obtained from the OR-Tool, for the optimization of the data set FT06. The values indicated in each cell are the Job that the machine performed and the Time, respectively ordered in this way, the result of the job on the left and the time on the right. This table contains six machines and with six task each, so then the tasks of the cells are the jobs and the distribution. Figure 5.1 presents the Gantt Chart which is a representation of Table 5.1. The Y axis presents the machines and the X axis the units of time needed for each work.

Machine0:	job_0_task_1	job_3_task_1	job_2_task_3	job_5_task_3	job_1_task_4	job_4_task_4
	[1,4]	[13,18]	[18,27]	[28,38]	[38,48]	[48,51]
Machine1:	job_1_task_0	job_3_task_0	job_5_task_0	job_0_task_2	job_4_task_1	job_2_task_4
	[0,8]	[8,13]	[13,16]	[16,22]	[22,25]	[27,28]
Machine2:	job_0_task_0	job_2_task_0	job_1_task_1	job_4_task_0	job_3_task_2	job_5_task_5
	[0,1]	[1,6]	[8,13]	[13,22]	[22,27]	[49,50]
Machine3:	job_2_task_1	job_5_task_1	job_0_task_3	job_3_task_3	job_1_task_5	job_4_task_5
	[6,10]	[16,19]	[22,29]	[29,32]	[48,52]	[52,53]
Machine4:	job_1_task_2	job_4_task_2	job_2_task_5	job_3_task_4	job_5_task_4	job_0_task_5
	[13,23]	[25,30]	[30,37]	[37,45]	[45,49]	**[49,55]**
Machine5:	job_2_task_2	job_5_task_2	job_1_task_3	job_0_task_4	job_4_task_3	job_3_task_5
	[10,18]	[19,28]	[28,38]	[38,41]	[41,45]	[45,54]

Table 5.1: Results using Google OR-Tool for optimization. Data set ft06.

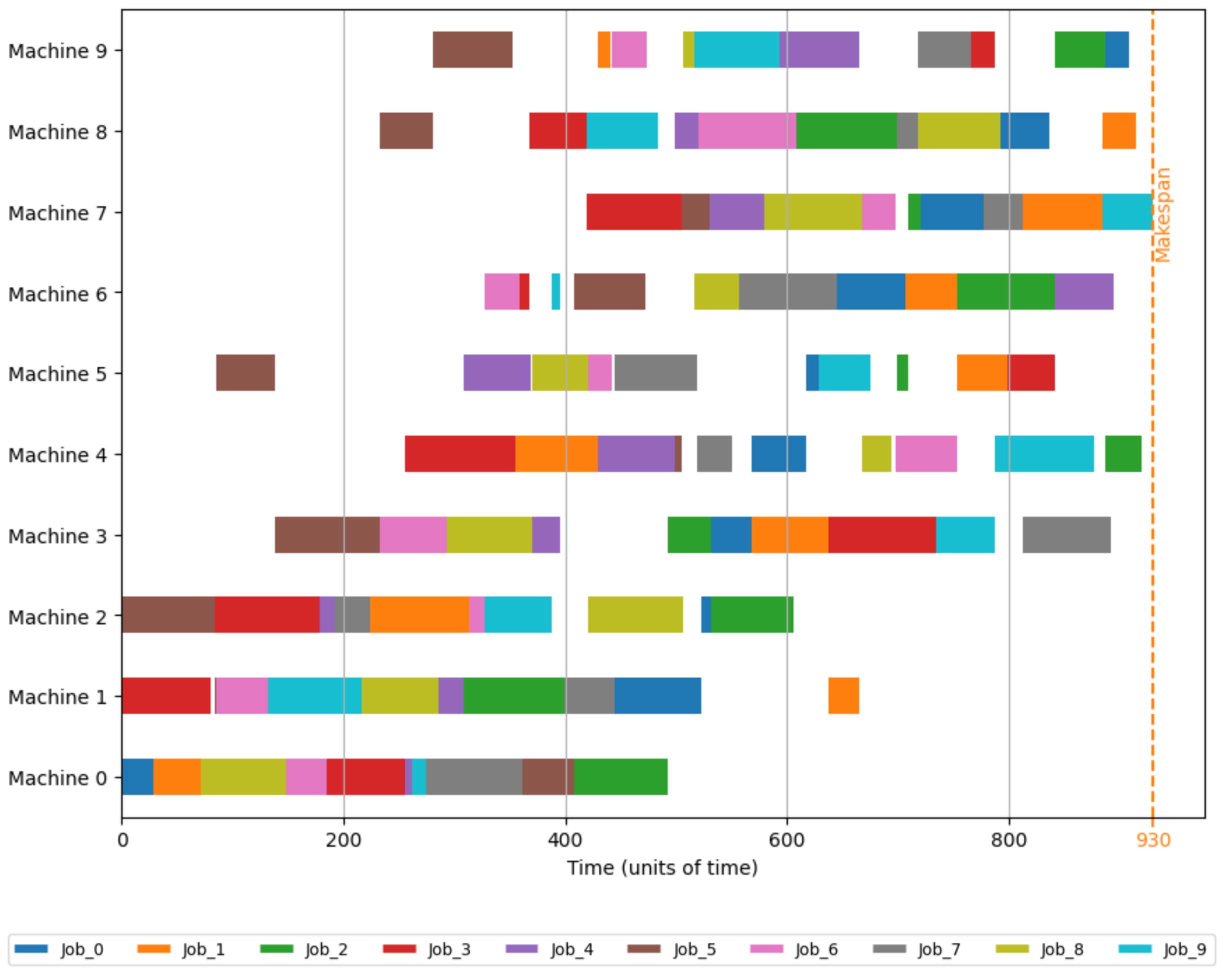


Figure 5.2: Serial Simulation with Google OR-Tool. ft10 Thomson and Fisher Data-set.

Figure 5.2 presents a Gantt Chart for the ft10 data set with the OR-Tool. The Y axis describes the data of the Machines, while the X axis represents the unit of time; this diagram shows in conjunction with the Jobs inside. Gantt Chart represents the workflow and schedule for each machine. The following colors guide the respective jobs and distribution. The color distribution will depend on every graph and even in every machine, and its function is to illustrate the final distribution.

Machine0:	job_0_task_0	job_1_task_0	job_8_task_0	job_6_task_1	job_3_task_2	job_4_task_1	job_9_task_1	job_7_task_1	job_5_task_6	job_2_task_1
	[0,29]	[29,72]	[72,148]	[148,185]	[185,256]	[256,262]	[262,275]	[275,361]	[361,408]	[408,493]
Machine1:	job_3_task_0	job_5_task_1	job_6_task_0	job_9_task_0	job_8_task_1	job_4_task_2	job_2_task_0	job_7_task_2	job_0_task_1	job_1_task_5
	[0,81]	[84,86]	[86,132]	[132,217]	[217,286]	[286,308]	[308,399]	[399,445]	[445,523]	[637,665]
Machine2:	job_5_task_0	job_3_task_1	job_4_task_0	job_7_task_0	job_1_task_1	job_6_task_3	job_9_task_2	job_8_task_4	job_0_task_2	job_2_task_3
	[0,84]	[84,179]	[179,193]	[193,224]	[224,314]	[314,327]	[327,388]	[421,506]	[523,532]	[532,606]
Machine3:	job_5_task_3	job_6_task_2	job_8_task_2	job_4_task_4	job_2_task_2	job_0_task_3	job_1_task_4	job_3_task_7	job_9_task_7	job_7_task_9
	[138,233]	[233,294]	[294,370]	[370,396]	[493,532]	[532,568]	[568,637]	[637,735]	[735,787]	[813,892]
Machine4:	job_3_task_3	job_1_task_2	job_4_task_5	job_5_task_8	job_7_task_4	job_0_task_4	job_8_task_8	job_6_task_9	job_9_task_8	job_2_task_9
	[256,355]	[355,430]	[430,499]	[499,505]	[519,551]	[568,617]	[668,694]	[698,753]	[787,877]	[887,920]
Machine5:	job_5_task_2	job_4_task_3	job_8_task_3	job_6_task_5	job_7_task_3	job_0_task_5	job_9_task_6	job_2_task_5	job_1_task_7	job_3_task_9
	[86,138]	[308,369]	[370,421]	[421,442]	[445,519]	[617,628]	[628,675]	[699,709]	[753,799]	[799,842]
Machine6:	job_6_task_4	job_3_task_4	job_9_task_3	job_5_task_7	job_8_task_6	job_7_task_5	job_0_task_6	job_1_task_6	job_2_task_7	job_4_task_9
	[327,359]	[359,368]	[388,395]	[408,473]	[517,557]	[557,645]	[645,707]	[707,753]	[753,842]	[842,895]
Machine7:	job_3_task_6	job_5_task_9	job_4_task_7	job_8_task_7	job_6_task_8	job_2_task_6	job_0_task_7	job_7_task_8	job_1_task_8	job_9_task_9
	[420,505]	[505,530]	[530,579]	[579,668]	[668,698]	[709,721]	[721,777]	[777,813]	[813,885]	**[885,930]**
Machine8:	job_5_task_4	job_3_task_5	job_9_task_4	job_4_task_6	job_6_task_7	job_2_task_4	job_7_task_6	job_8_task_9	job_0_task_8	job_1_task_9
	[233,281]	[368,420]	[420,484]	[499,520]	[520,609]	[609,699]	[699,718]	[718,792]	[792,836]	[885,915]
Machine9:	job_5_task_5	job_1_task_3	job_6_task_6	job_8_task_5	job_9_task_5	job_4_task_8	job_7_task_7	job_3_task_8	job_2_task_8	job_0_task_9
	[281,353]	[430,441]	[442,474]	[506,517]	[517,593]	[593,665]	[718,766]	[766,788]	[842,887]	[887,908]

Table 5.2: Results using Google OR-Tool for optimization. Data set ft10.

Table 5.2 describes the results of the OR-Tool using the ft10 Data set. The tasks in the cells represent the job and task id. Each job has the start date and the end date in units of time. The Table 5.2 shows in numbers the Figure 5.2.

Figure 5.3 represents the ft20 benchmark data set optimized by OR-Tool. The Y axis represents the five machines of the data, and the X axis represents the units of time to finish in a shorter time. The colors present the jobs are detailed in the legend of the figure. Finally, Table 6 illustrates in detail the job and task id and their values. The Table 6 is in the Appendix .2 in order to show it better due to the long size of the table. Those values are the optimal obtained by OR-Tool showed in the Figure 5.3. It is important to remark that the table is in the Appendix Section .2 due to the large space needed.

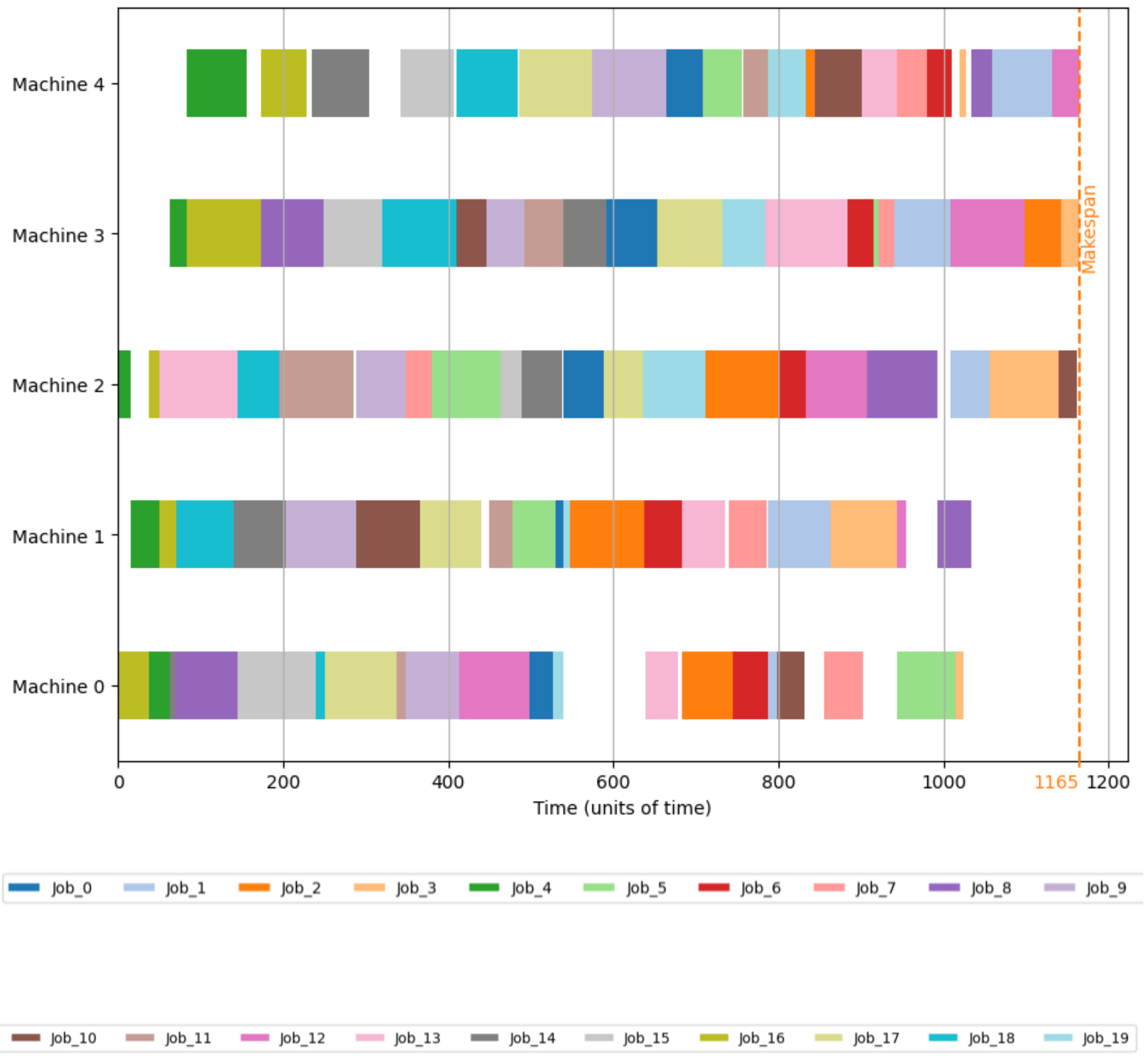


Figure 5.3: Serial Simulation with Google OR-Tool. ft20 Thomson and Fisher Data-set.

5.2 DES Implementation

This section represents the results of the Discrete Event Simulation implementation. The benchmark data sets in this method are ft06, ft10, and ft20. The DES implementation follows the model constraints and items specified in Chapter 4. The figures include legends and colors to identify the sequence of the jobs. The tables also include more values to manage the job and task id easily as data for each job.

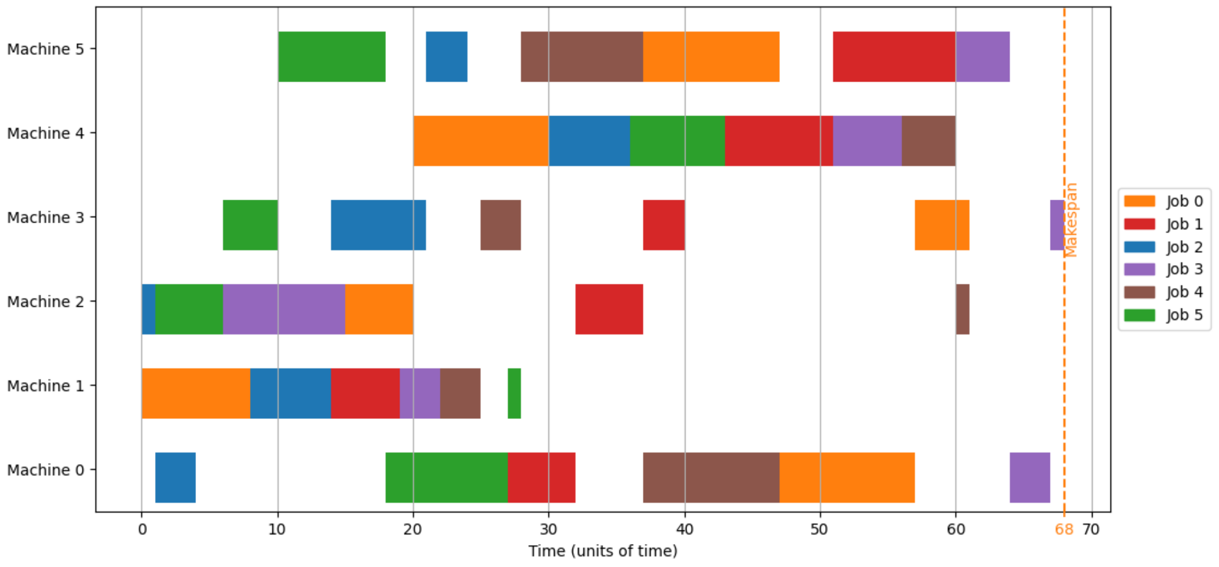


Figure 5.4: Serial Simulation with DES Model ft06 Thomson and Fisher Data-set.

Machine0:	job_0_task_1	job_2_task_3	job_3_task_1	job_5_task_3	job_1_task_4	job_4_task_4
	0,1,1,4	2,3,18,27	3,1,27,32	5,3,37,47	1,4,47,57	4,4,64,67
Machine1:	job_1_task_0	job_0_task_2	job_3_task_0	job_4_task_1	job_5_task_0	job_2_task_4
	1,0,0,8	0,2,8,14	3,0,14,19	4,1,19,22	5,0,22,25	2,4,27,28
Machine2:	job_0_task_0	job_2_task_0	job_4_task_0	job_1_task_1	job_3_task_2	job_5_task_5
	0,0,0,1	2,0,1,6	4,0,6,15	1,1,15,20	3,2,32,37	5,5,60,61
Machine3:	job_2_task_1	job_0_task_3	job_5_task_1	job_3_task_3	job_1_task_5	job_4_task_5
	2,1,6,10	0,3,14,21	5,1,25,28	3,3,37,40	1,5,57,61	**4,5,67,68**
Machine4:	job_1_task_2	job_0_task_5	job_2_task_5	job_3_task_4	job_4_task_2	job_5_task_4
	1,2,20,30	0,5,30,36	2,5,36,43	3,4,43,51	4,2,51,56	5,4,56,60
Machine5:	job_2_task_2	job_0_task_4	job_5_task_2	job_1_task_3	job_3_task_5	job_4_task_3
	2,2,10,18	0,4,21,24	5,2,28,37	1,3,37,47	3,5,51,60	4,3,60,64

Table 5.3: Results using Discrete Event Simulation development model. Data set ft06.

In Table 5.3, the obtained machines' values are represented by: Job, Task, Start, and End. For a better understanding, one of the values of the machines is the following. Thus, Machine 0 (First column) and Job 0 Task 1 (Second column) gives the following values: 0, 1, 1, 4. In this way, the first value represents the Job; the second value represents the Task, the third value represents the Start, and finally, the last value represents the End. Figure 5.4 represents the table graphically using the Gantt chart to provide an easy way to understand the result.

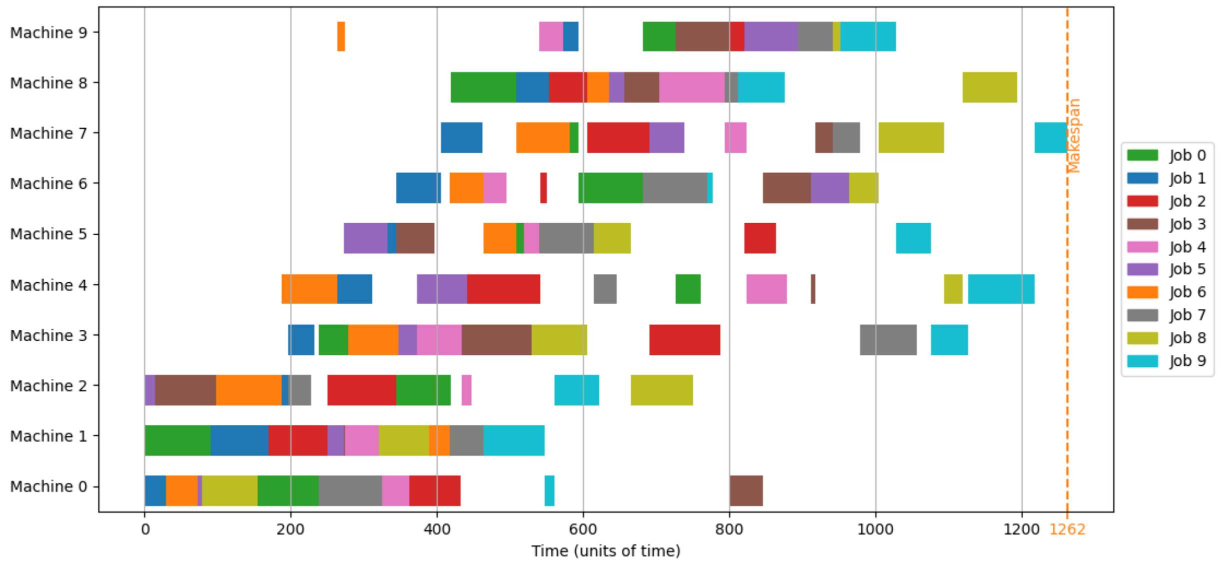


Figure 5.5: Serial Simulation with DES Model ft10 Thomson and Fisher Data-set.

Machine0:	job_0_task_0	job_1_task_0	job_4_task_1	job_8_task_0	job_2_task_1	job_7_task_1	job_6_task_1	job_3_task_2	job_9_task_1	job_5_task_6
	0, 0, 0, 29	1, 0, 29, 72	4, 1, 72, 78	8, 0, 78, 154	2, 1, 154, 239	7, 1, 239, 325	6, 1, 325, 362	3, 2, 362, 433	9, 1, 548, 561	5, 6, 799, 846
Machine1:	job_2_task_0	job_0_task_1	job_3_task_0	job_4_task_2	job_5_task_1	job_6_task_0	job_8_task_1	job_1_task_5	job_7_task_2	job_9_task_0
	2, 0, 0, 91	0, 1, 91, 169	3, 0, 169, 250	4, 2, 250, 272	5, 1, 272, 274	6, 0, 274, 320	8, 1, 320, 389	1, 5, 389, 417	7, 2, 417, 463	9, 0, 463, 548
Machine2:	job_4_task_0	job_5_task_0	job_1_task_1	job_0_task_2	job_7_task_0	job_3_task_1	job_2_task_3	job_6_task_3	job_9_task_2	job_8_task_4
	4, 0, 0, 14	5, 0, 14, 98	1, 1, 98, 188	0, 2, 188, 197	7, 0, 197, 228	3, 1, 250, 345	2, 3, 345, 419	6, 3, 434, 447	9, 2, 561, 622	8, 4, 665, 750
Machine3:	job_0_task_3	job_2_task_2	job_1_task_4	job_4_task_4	job_6_task_2	job_5_task_3	job_8_task_2	job_3_task_7	job_7_task_9	job_9_task_7
	0, 3, 197, 233	2, 2, 239, 278	1, 4, 278, 347	4, 4, 347, 373	6, 2, 373, 434	5, 3, 434, 529	8, 2, 529, 605	3, 7, 690, 788	7, 9, 978, 1057	9, 7, 1075, 1127
Machine4:	job_1_task_2	job_0_task_4	job_4_task_5	job_3_task_3	job_7_task_4	job_2_task_9	job_6_task_9	job_5_task_8	job_8_task_8	job_9_task_8
	1, 2, 188, 263	0, 4, 263, 312	4, 5, 373, 442	3, 3, 442, 541	7, 4, 614, 646	2, 9, 727, 760	6, 9, 823, 878	5, 8, 911, 917	8, 8, 1093, 1119	9, 8, 1127, 1217
Machine5:	job_4_task_3	job_0_task_5	job_5_task_2	job_1_task_7	job_2_task_5	job_6_task_5	job_7_task_3	job_8_task_3	job_3_task_9	job_9_task_6
	4, 3, 272, 333	0, 5, 333, 344	5, 2, 344, 396	1, 7, 463, 509	2, 5, 509, 519	6, 5, 519, 540	7, 3, 540, 614	8, 3, 614, 665	3, 9, 821, 864	9, 6, 1028, 1075
Machine6:	job_0_task_6	job_1_task_6	job_6_task_4	job_3_task_4	job_2_task_7	job_7_task_5	job_9_task_3	job_5_task_7	job_4_task_9	job_8_task_6
	0, 6, 344, 406	1, 6, 417, 463	6, 4, 463, 495	3, 4, 541, 550	2, 7, 593, 682	7, 5, 682, 770	9, 3, 770, 777	5, 7, 846, 911	4, 9, 911, 964	8, 6, 964, 1004
Machine7:	job_0_task_7	job_1_task_8	job_2_task_6	job_3_task_6	job_4_task_7	job_6_task_8	job_5_task_9	job_7_task_8	job_8_task_7	job_9_task_9
	0, 7, 406, 462	1, 8, 509, 581	2, 6, 581, 593	3, 6, 605, 690	4, 7, 690, 739	6, 8, 793, 823	5, 9, 917, 942	7, 8, 942, 978	8, 7, 1004, 1093	**9, 9, 1217, 1262**
Machine8:	job_2_task_4	job_0_task_8	job_3_task_5	job_1_task_9	job_4_task_6	job_5_task_4	job_6_task_7	job_7_task_6	job_9_task_4	job_8_task_9
	2, 4, 419, 509	0, 8, 509, 553	3, 5, 553, 605	1, 9, 605, 635	4, 6, 635, 656	5, 4, 656, 704	6, 7, 704, 793	7, 6, 793, 812	9, 4, 812, 876	8, 9, 1119, 1193
Machine9:	job_1_task_3	job_6_task_6	job_0_task_9	job_2_task_8	job_5_task_5	job_3_task_8	job_4_task_8	job_7_task_7	job_8_task_5	job_9_task_5
	1, 3, 263, 274	6, 6, 540, 572	0, 9, 572, 593	2, 8, 682, 727	5, 5, 727, 799	3, 8, 799, 821	4, 8, 821, 893	7, 7, 893, 941	8, 5, 941, 952	9, 5, 952, 1028

Table 5.4: Results using Discrete Event Simulation development model. Data set ft10.

Figure 5.5 indicates a Gantt chart that shows the Job performed by ten machines (Y-Axis) in a certain period (X-Axis); thus, the Jobs follows the colors in the legend of the figure. Table 5.4 represents the results of the DES for the ft10 data set. To understand the values and logic of the table. The following example details the table results: Job, Task, Start, End. Thus, Machine0 (First column) and job_0_task_0 (Second column) give the following values: 0, 0, 0, 29. In this way, the first value represents the Job; the second value represents the Task, the third value represents the Start, and finally, the last value represents the End.

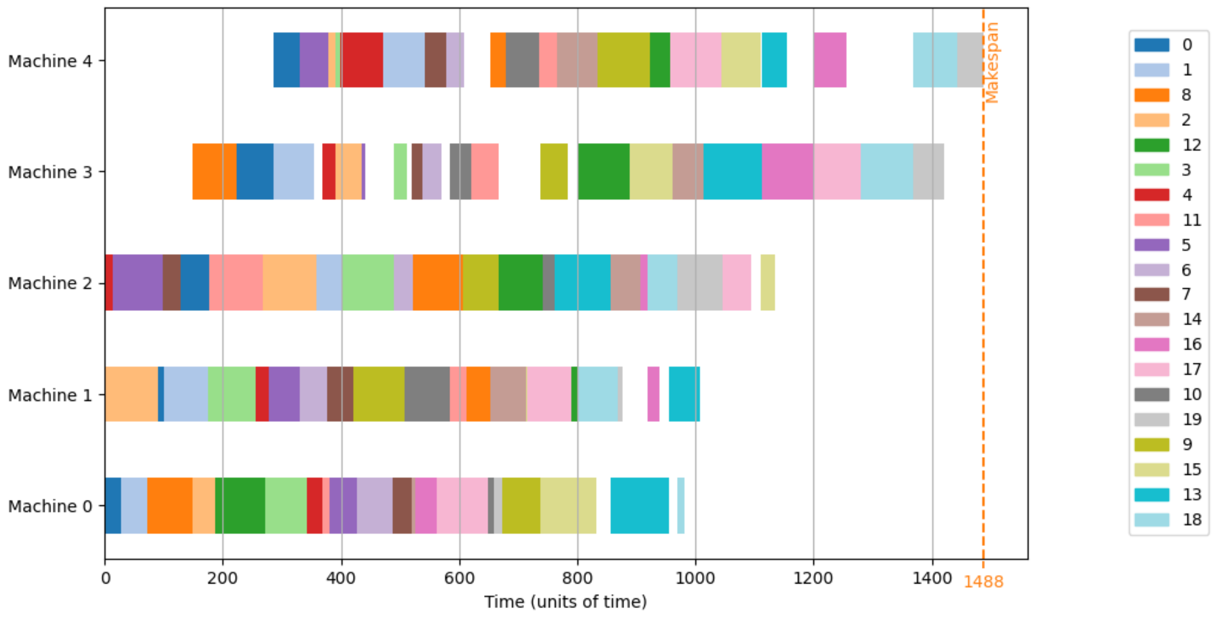


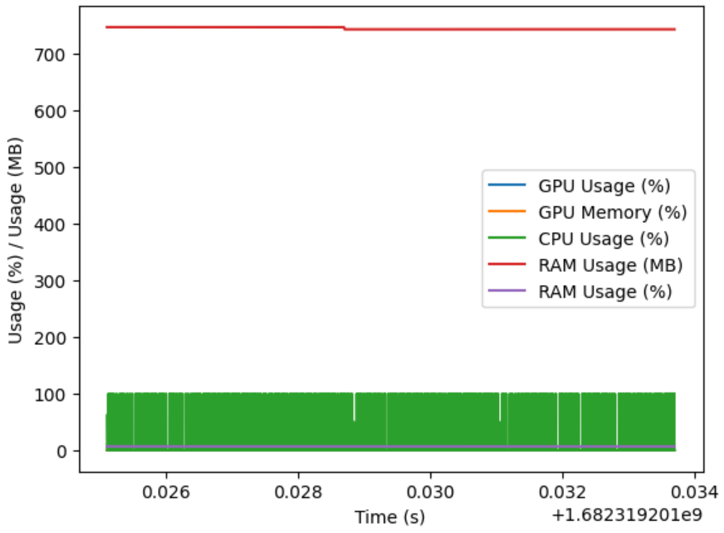
Figure 5.6: Serial Simulation with DES Model ft20 Thomson and Fisher Data-set.

Figure 5.6 shows the results of the Discrete Event Implementation Simulation. The logic of the chart follows the previous figures. The twenty jobs are in the five machines showing the simulate distribution. The colors in this implementation are different to show the best simulated distribution to any customer or administrator. The Table 7 represents the same results in a operational table. The jobs and tasks id are defined to explain the simulated process. The Table 7 is in the Appendix .2 in order to show it better due to the long size of the table. Similarly, Appendix .3 contains the long version of the DES implementation Gantt charts to visualize the results adequately. The results obtained were obtain in two scenarios, the first one in the Central Process Unit (CPU) or serial implementation and the second using Graphics Processing Unit (GPU) or parallel implementation.

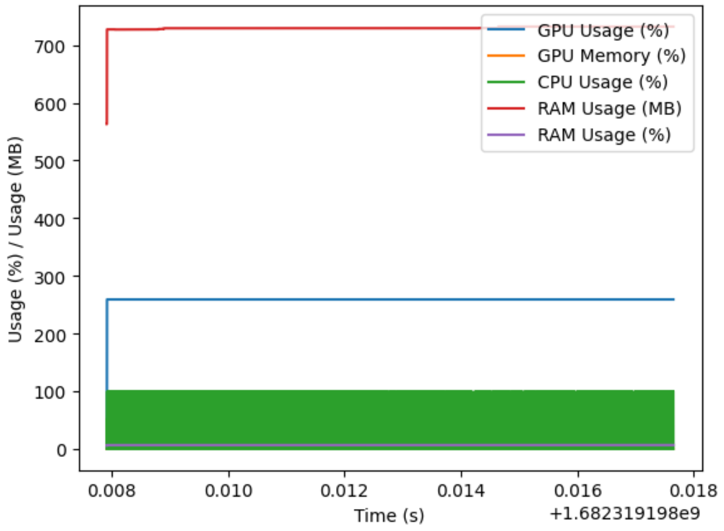
5.3 Parallel Implementation

This section illustrates the results of the parallel implementation. The ft20 data set is the benchmark data set due to the complexity of the jobs. The serial and parallel implementations compare the computational resources. The idea is to use a cluster, an High Performance Computing system, to compare the same code in serial and parallel implementation. The basis is to compare if the High Performance Computing system is a viable computational resource to achieve new results from the simulation.

One important observation is that the y-axis vary in each figure depending of MB or percentage % plotted. On the other hand, the x-axis indicates the time in seconds and each point illustrates the incremental of time during the method.



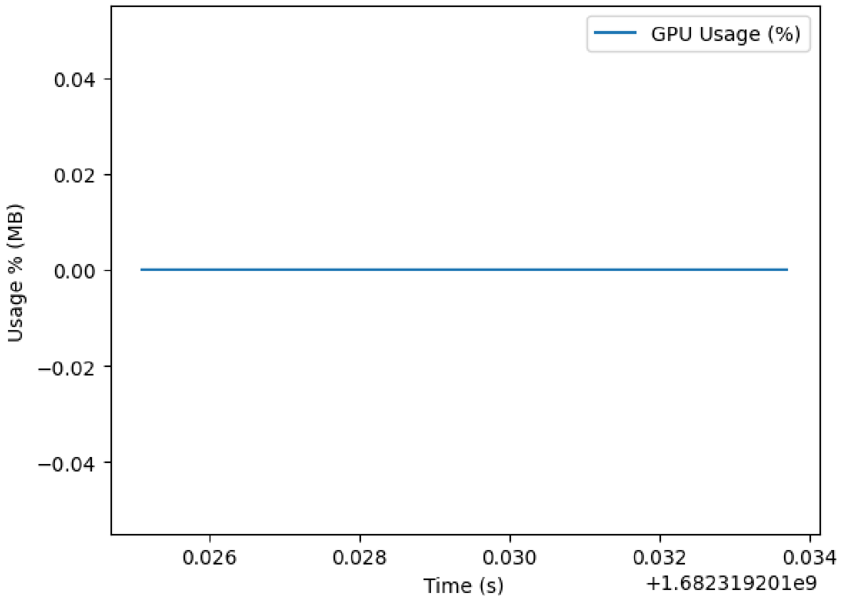
(a) CPU or Serial Implementation



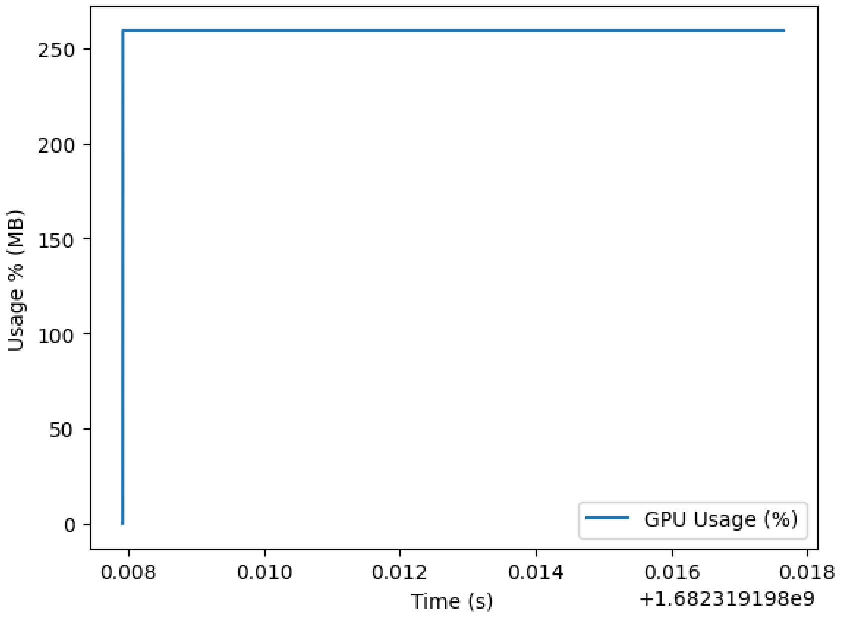
(b) GPU or Parallel Implementation

Figure 5.7: Overall Overview of the Computational Resource Consumption

Figure 5.7 communicates the metrics obtained between the CPU or serial implementation and GPU or parallel. The measures are in percentage and MB used. The metrics involve GPU, CPU, and RAM analysis. The cluster used the same computational conditions with the variation in the compute logic. The graphics display the metrics in lines, just CPU Usage shows a continuous graph.



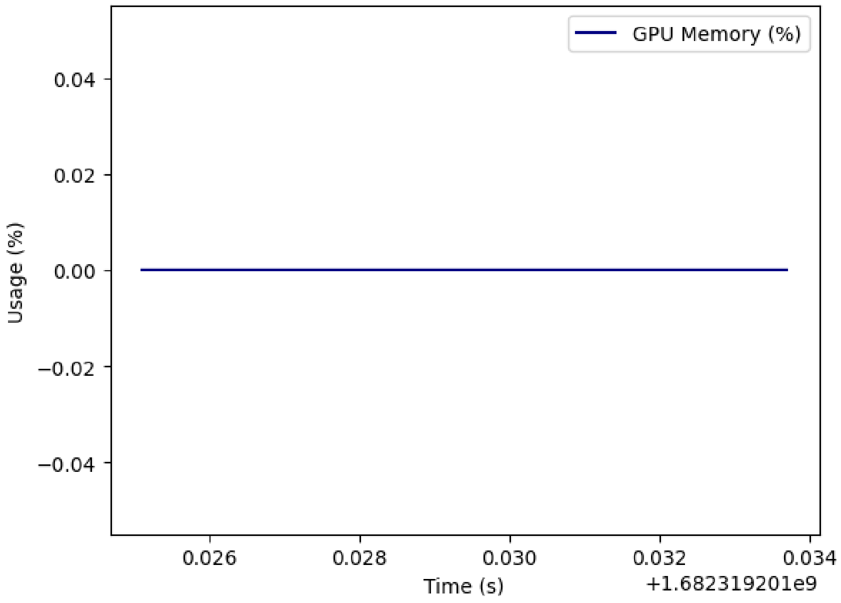
(a) CPU or Serial Implementation



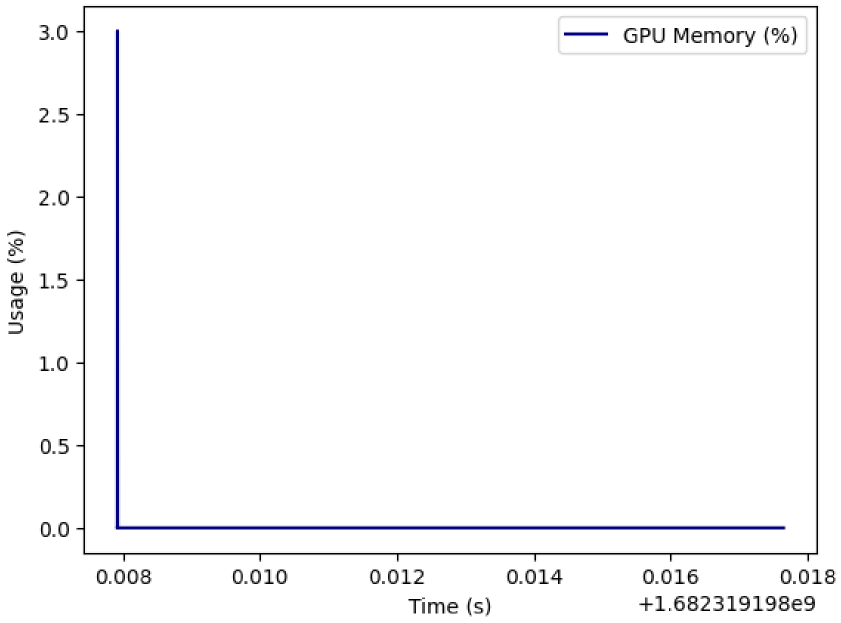
(b) GPU or Parallel Implementation

Figure 5.8: GPU Usage in MB during the process.

Figure 5.8 represents the percentage of MB used along the time in the process. It implies the execution of one or more kernels in the Graphics Processing Unit. This is the only metric that the sign of % implies the percentage usage in MB. Other cases implies percentage only.



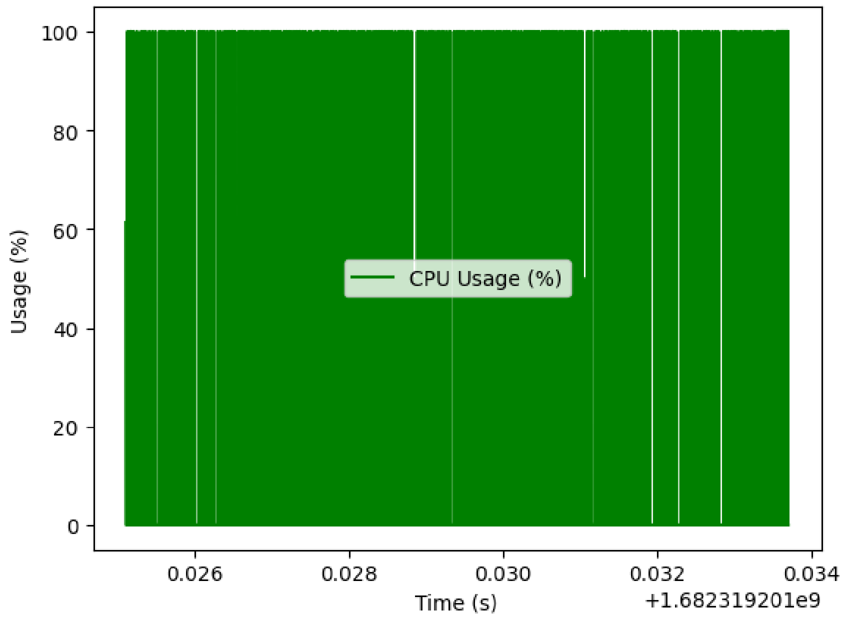
(a) CPU or Serial Implementation



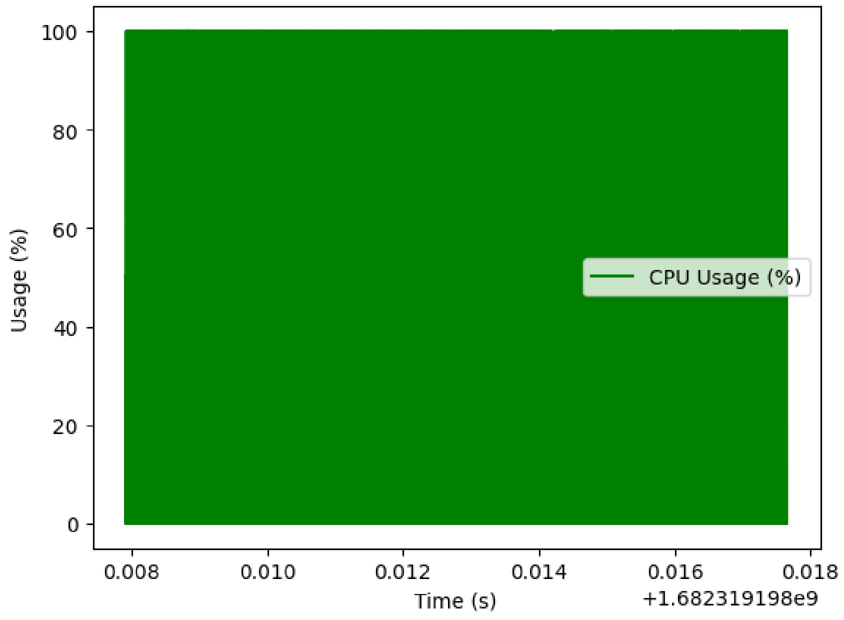
(b) GPU or Parallel Implementation

Figure 5.9: GPU Memory in percentage during the process.

Figure 5.9 presents in percentage the GPU Usage. It represents the usage percentage in the VRAM of the GPU. The figures represent the changes over time in the use of resources. Each peak means the max value over time, which keeps constant until another peak value can exceed it. The zero value implies that no new value can exceed the previously established one.



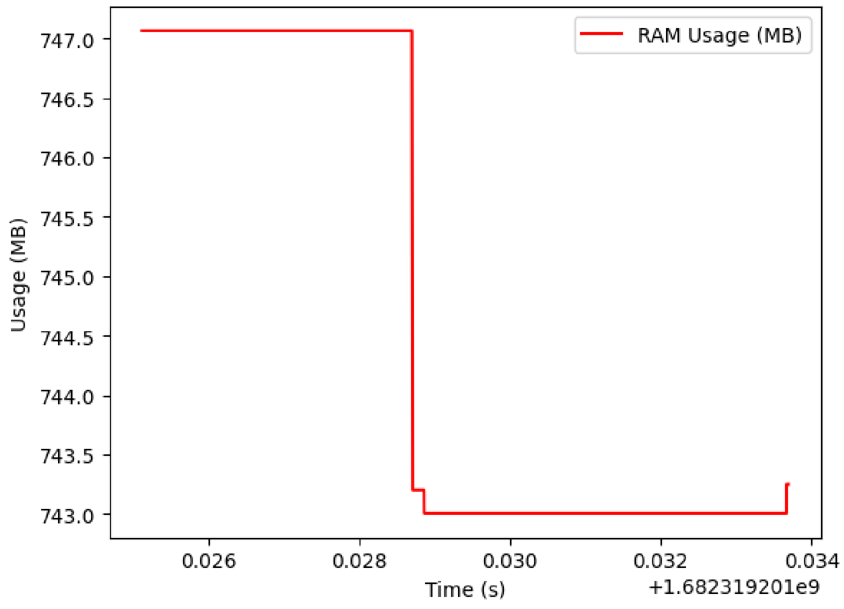
(a) CPU or Serial Implementation



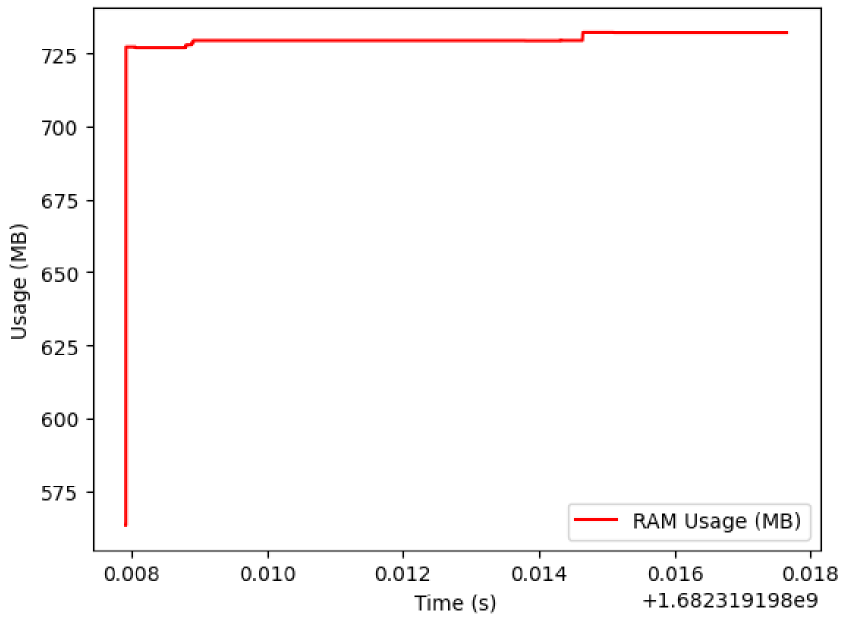
(b) GPU or Parallel Implementation

Figure 5.10: CPU Usage in percentage during the process.

Figure 5.10 refers to the CPU usage over time. The maximum value is one hundred percent in each instance of time. In order to illustrate the CPU usage, the graph will keep the values for a long time. The CPU usage metric is the unique metric that keeps the value over time to illustrate the most useful metric.



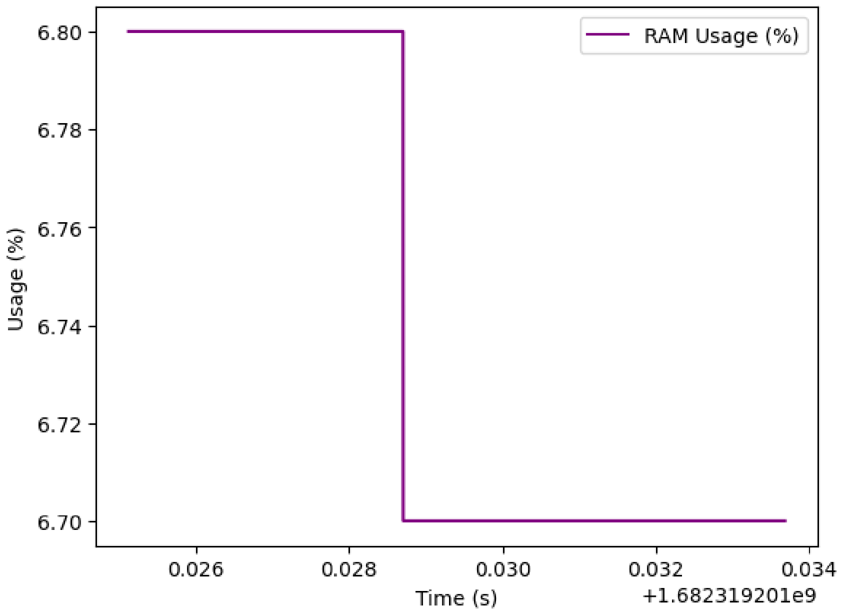
(a) CPU or Serial Implementation



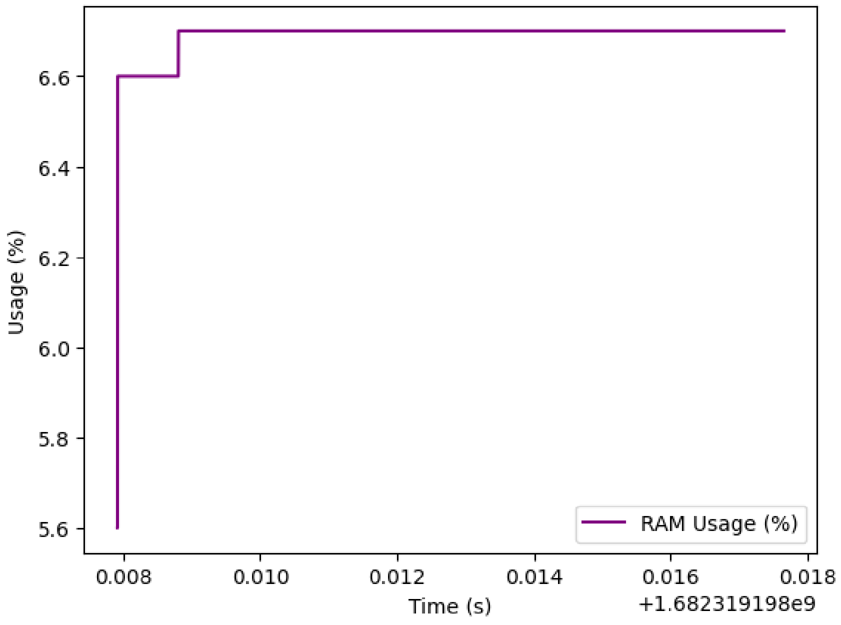
(b) GPU or Parallel Implementation

Figure 5.11: RAM Usage in MB during the process.

Figure 5.11 plots the RAM usage along time during the serial and parallel implementation. The value indicates the number in MB used during the operation. The figures compare the serial and parallel execution against the RAM usage metric.



(a) CPU or Serial Implementation



(b) GPU or Parallel Implementation

Figure 5.12: RAM Usage in percentage during the process.

Figure 5.12 emphasizes in percentage the use of RAM during the operation. The figure analyzes the two implementations and the percentage used in general of the resources. It measures the percentage proceeds calculated concerning all the cluster results.

Chapter 6

Discussion

This chapter describes in detail the results obtained in the workflow explained in the previous chapter. Significantly, the main concerns of the serial implementation comparison between the OR-Tool and Discrete Event Simulation Implementation. Besides, the comparison between the computational resources in the serial and the parallel implementation in the High-Performance Computing system.

6.1 Serial Implementation

During this section, the FT06, FT10, and FT20 are benchmark data sets configured in OR-Tools and Discrete Event Simulation implementation. First, it is essential to mention that Google OR-Tool has been the best optimization tool since 2013, obtaining the gold medal in the International Constraint Programming Competition every year until now. On the other hand, the Discrete Event Simulation (DES) presents a more accessible and flexible implementation for companies with medium and low technological resources. In order to understand each graphic, we consider that to complete a Job; this Job must go through all the Machines to obtain a final result. Like real scenarios, the raw material brings a product through manufacturing. The Gantt charts explain the process of each machine, coordinating the necessary steps and requirements.

Comparing the serial implementations, the Figure 5.2 and Table 5.1 related to the OR-Tools implementation with the ft06 data set shows a better make-span optimization value of 55 units of time. Using this tool, Machine 0 completed its workflow in 51 units of time, Machine 1 in 28 units of time, Machine 2 in 50 units of time, Machine 3 in 53 units of time, Machine 4 in 55 units of time, and finally, the Machine 5 in 54 units of time. In contrast, the Discrete Event Simulation for the ft06 in Figure 5.4 and Table 5.3 shows a makespan optimization value of 68. The total time in each machine is Machine 0 in 67 units of time, Machine 1 in 28 units of time, Machine 2 in 61 units of time, Machine 3 in 68 units of time, Machine 4 in 60 units of time, and Machine 5 in 64 units of time. Only in Machine 1 the time is the same; otherwise, OR-Tools shows a clear improvement compared with DES. Another characteristic during the process flow in Machines 2 and 3 is similar, changing the positions of a couple of jobs and tasks. For example, Machine 2 has the sequence in OR-Tools job_0_task_0, job_2_task_0, job_1_task_1, job_4_task_0, job_3_task_2 and job_5_task_5. Moreover, DES has the sequence job_0_task_0, job_2_task_0, job_4_task_0, job_1_task_1, job_3_task_2, job_5_task_5. It shows a minimal change between the third and fourth steps in the process. Nevertheless, the small changes in those machines and the biggest in others imply a gap of 13 units of time between OR-Tools and DES using the ft06 data set. The percentage error is 23.63% considering the OR-Tools as the best implementation.

On the other hand, using the benchmark ft10, Figure 5.2 and Table 5.2 represent the OR-Tools implementation. This method obtained the value for the makespan function of 930 units of time (ut). Since this time is by convenience, the term ut will represent the units of time. In the OR-Tools method, the results per machine are the follows, Machine 0 finishes in 493 ut, Machine 1 in 665 ut, Machine 2 in 606 ut, Machine 3 in 892 ut, Machine 4 in 920 ut, Machine 5 in 842 ut, Machine 6 in 895 ut, Machine 7 in 930 ut, Machine 8 in 915 ut, Machine 9 in 908 ut. On the opposite method, Figure 5.5 and Table 5.4 represent the Discrete Event Simulation implementation with a makespan optimization result of 1262 ut. Thus, analyzing each machine, the results are Machine 0 finishes in 846 ut, Machine 1 in 548 ut, Machine 2 in 750 ut, Machine 3 in 1127 ut, Machine 4 in 1217 ut, Machine 5 in 1075 ut, Machine 6 in 1004 ut, Machine 7 in 1262 ut, Machine 8 in 1193 ut, Machine 9 in 1028 ut. To sum up, OR-Tools, on average, improves the DES times. Even one-by-one case OR-Tools keep a significant improvement. It is valuable to show that in the case of Machine 0, DES implementation almost has twice the value in time units. Mainly for this data set, the ft10 DES workflow differs from the OR-Tools implementation. Thus, there is a clear trend in the increase of variables and the simulated model complexity compared to the previous case. This increase in computational complexity is significant, which implies less accuracy in the DES model. Due to the exponential increase in tasks to be solved, the DES model has a 332 ut gap concerning the best optimization, a deteriorated result due to the increase in complexity compared to its previous performance. The percentage error is 35.69% considering the OR-Tools as the best implementation. In brief, if more complex is the industrial model more difficult for the DES implementation to find the optimal solution. Other methods, such as OR-Tools and Cpmodel, could achieve the optimal value. It is also important to mention that DES implementation has fewer years of implementation, which can justify its performance related to the OR-Tools, which have many years of development.

Finally, the last comparison between the DES and OR-Tools implementation is in the ft20 data set. Figure 5.3 and Table 6 show the OR-Tools implementation. The Table 6 is in the Appendix .2 in order to show it better due to the long size of the table. While Figure 5.6 and Table 7 illustrate the DES implementation results. The Table 7 is in the Appendix .2 in order to show it better due to the long size of the table. The DES implementation achieves 1488 units of time to finish the workflow. However, the OR-Tools implementation achieves the 1164 units of time to complete this process with this amount of machines and jobs. The absolute error is 324, and the calculated percentage error is 27.83 %. Analyzing item by item, the following results are Machine 0 in 1015 ut, Machine 1 in 1033 in ut, Machine 2 in 1161, Machine 3 in 1165, and Machine 4 in 1164 using OR-Tools. The DES implementation results are Machine 0 in 981 ut, 1 in 1008 in ut, 2 in 1135, 3 in 1421, and 4 in 1488 using OR-Tools. The main contrast with the other benchmarks process is decent in the error percentage. Compared with the ft10 implementation, the ft20 implementation has a better performance. It means that increasing the number of jobs and tasks does not increase the complexity as quickly as increasing the number of machines. Even the Machine 0 improves the times with the DES compared with OR-Tools. However, in the set of solutions the OR-Tools has a better optimization times.

It's a clear conclusion about the most accurate tool, with a difference, is the OR-Tool Implementation. The DES errors are in ft06 of 23.63%, ft10 of 35.69%, and ft20 of 27.83

% compared with OR-Tools. The OR-Tools code is the best optimal tool know until now. Since 2013, the OR-Tools is the best optimizer. Thus, the importance to compare the DES implementation with the best tool to improve the DES details and computational logic. Due to the short implementation time, DES has a range to improve over the time. Mainly to foccuss on the scalability drawback for medium and low-growth companies. Therefore, the graphs of DES implementation has a better explanation for the manufacturing schedule implementation and the tables generated have more information to scale over time. Since DES is a basic tool in this stage it illustrates potential configuration modeling in order to provide a layer of modification to fit the companies requirements.

6.2 Parallel Implementation

Using parallelization tools in this problem is justified due to the number of existing variables. It is also important to emphasize that job shop scheduling optimization with any algorithm is considered a complex issue. In this work, the HPC system aims to decrease the time and shortly uses it as a tool to obtain better results for complex models. Another consideration for using HPC is lengthening the proposed process and managing to increase the number of machinery. The proposed environment for the HPC is to use the DES in the serial version and the DES in the parallel version, both with the ft20 benchmark data set. This implementation seeks to test which are the heaviest sections of the production to implementation. Thus, check the HPC alternative for this critical section.

For the comparison, the FT10 data set used serial or CPU resources against GPU or parallel resources in the DES implementation. It activated a GPU kernel in parallel mode to take advantage of the resources. Thus, Figure 5.7 shows the computational metrics obtained during this comparison. It reveals the consumption time of the resources and the percentage of their consumption. First of all, the main difference is the execution time. The serial version increments time compare with the parallel execution. The wide variation of time is 0.015999403 seconds. The high-level characteristics of the HPC system reduced the execution comparison time. Hence the importance to incorporate those systems to execute the industrial scheduling planning. In the serial implementation, the simulation time was 8.659337043762207 seconds using the local computer, a powerful contrast with the 1.700319798 seconds in the HPC structure.

Figure 5.8 interprets the GPU usage in MB along the time in both techniques. The serial method did not use the GPU resources; in contrast, the parallel method used 259.125 MB during the execution time. It illustrates the parallel aim of using GPU aids. Figure 5.9 identifies the GPU leap in the parallel implementation. Subfigure (b) shows the jumps in percentage % of the peaks in the GPU use. It maintains the 3% peak over time; no variation affects this metric since no peak is over 3%. Subfigure (a) did not denote changes because this metric was not activated.

In the same way, Figure 5.10 outlines CPU percentage usage. The main characteristic is that parallel implementation keeps the same value over time. By contrast, the serial method has some instances with a lower CPU percentage of usage over time. This performance variation relies on an optimization use of resources. The previous graph denotes that even the parallel version uses more GPU and CPU resources, computational parameters which are strongly related to time decreasing.

Figure 5.11 highlights the RAM usage during the process. Subfigure (a) tendency suggests a decrease from 747.06640625 MB to 743.00390625 MB. Conversely, subfigure (b) came from 563.2695313 MB to 732.140625 MB. Even though parallel activates a new GPU kernel, the serial implementation maintains the higher values. The slight decrease over time reflects the difficulty of maintaining high-level resources even when the capability of the cluster is higher than parallel. It suggests that the small changes in the CPU did not perform the scheduling or the parallel code. Analogous to RAM Usage (MB), Figure 5.12 represents the same metric in terms of percentage. Since the cluster maximum resources are 100 %, both serial and parallel are far away to achieve all the computational power. The RAM consumption in percentage significantly drops in the serial part in the middle of the process. By the way, using more percentage of the resources between 6.8 % and 6.7 %, the serial did not achieve less time. Nevertheless, the parallel implementation remained at a regular percentage of 6.6 % over time, which means that the simulation implementation obtained more acceptable results with constant use of resources.

Chapter 7

Conclusions

7.1 Performance of the method

In conclusion, this study sought to determine a viable simulation procedure for scheduling the manufacturing process. Through the use of Discrete Event Simulation; we designed, implemented, and tested a simulation engine. The simulation aims to provide a scheduling solution for more complex scenarios. Supported by the data sets ft06, ft10, and ft20, we found that by increasing the number of variables, machines, and jobs, the complexity of the simulation correlated directly. One main contribution related to data scalability is the detection of consistent performance while adding jobs; in contrast, adding machines implies to increase faster complexity. After comparing with the best optimization tool at this time, OR-Tools powered by Google. DES has a reasonable range of errors considering the novel implementation of this tool. These findings support the growing model under the demand of the manufacturing industry and suggest a viable method under industrial conditions. Finally, this study provides a comparison using High-Performance Computing, finding that the incremental computing resources have no incremental benefits. Using GPU resources, we found that the main contribution is time reduction, even using less Random Access Memory. Thus, for future near contributions, applying real scenario extensive data is needed to increment the potential optimization in the method. Future research can investigate the implications of large real scenarios to seek Digital Twin for medium and small low-automated industries.

7.2 Evaluation of the objectives

The workflow indeed achieves a collection of the environment variables and model constraints related to the manufacturing procedure. It defined the simulation engine structure and methodology. It alludes to recreating the Discrete Event Simulation implementation. It fits the approximation to the manufacturing strategy and returns consistent values related to the administration planning of resources. Hence, this work aims at the initial specific objective.

Then, the Discrete Event Simulation method was executed and tested with the most helpful optimization tool. OR-Tools represented the best possible optimization, obtaining a starting point to compare results and graphical methods. Then, Gantt Chart is the visualization tool implemented in both methods. This visualization tool allows us to obtain, at first glance, the entire simulation in an interactive and easy-to-understand manufacturing workflow. In this sense, we achieved the second specific objective.

Finally, incorporating the High-Performance Computing system, the Discrete Event Simulation tested the computer metrics to find alternatives to this method. The computational metrics compare the conventional and parallel approaches to take advantage of the computational advances. We were obtaining critical points since the comparison. Hence, we completed the third specific objective defined in this work.

In brief, the three-phase methodology that we implemented in this research. Using the benchmark data sets ft06, ft10, and ft20 to compare with the best optimization tool OR-Tools; allowed us to recreate a Discrete Event Simulation engine for manufacturing optimization. Then, the HPC system provided us with the compare metrics to analyze novel implementing alternatives. In this way, we achieved the general objective of this work.

7.3 Future Work

7.3.1 GUI

For future advancement in this work, it is important to remark that the interaction between the user and the program, for which a clear path is the creation of a graphical interface that allows the user not only to modify the model but also to set parameters that fit their physical environment. It is important to emphasize that although this implementation is through code, a much simpler way for interaction with an end user would be to present it graphically.

7.3.2 High Performance Computing

In the high-performance computing section, a clear advance is the use of better mechanisms such as more advanced clusters, a very good association for this future work would be the implementation of this work in CEDIA, which implies a great challenge for the fact that being able to import the libraries and work with objects to another programming language. However, it is worth emphasizing that more advanced computer systems could yield more favorable results for this work.

7.3.3 Data

In the near future, obtaining data from a low-automation industry will improve the discrete event simulation model accuracy. To summarize, modeling real scenarios will incorporate new challenges and solutions to the current project. Besides, it is essential to remark that Ecuadorian companies do not share the data for research projects, implying a risk in improving the model shortly. Furthermore, the no automated industry data will include novel features to the model. Thus, it the importance to incorporate new data. In contrast to the automated industry data, the data based on low automatization will support a new vision required for Latin America Industry development.

Bibliography

[1] W. Li, B. H. Huynh, H. Akhtar, and K. S. Myo, "Discrete event simulation as a robust supporting tool for smart manufacturing," in *Implementing Industry 4.0*. Springer, 2021, pp. 287–312.

[2] D.-Y. Jeong, M.-S. Baek, T.-B. Lim, Y.-W. Kim, S.-H. Kim, Y.-T. Lee, W.-S. Jung, and I.-B. Lee, "Digital twin: Technology evolution stages and implementation layers with technology elements," *IEEE Access*, 2022.

[3] V. Kindratenko and P. Trancoso, "Trends in high-performance computing," *Computing in Science & Engineering*, vol. 13, no. 3, pp. 92–95, 2011.

[4] A. Tekin, A. Tuncer Durak, C. Piechurski, D. Kaliszan, F. Aylin Sungur, F. Robertsén, and P. Gschwandtner, "State-of-the-art and trends for computing and interconnect network solutions for hpc and ai," *Partnership for Advanced Computing in Europe, Available online at www. praceri. eu*, 2021.

[5] Z. Bi, Y. Jin, P. Maropoulos, W.-J. Zhang, and L. Wang, "Internet of things (iot) and big data analytics (bda) for digital manufacturing (dm)," *International Journal of Production Research*, pp. 1–18, 2021.

[6] K. Alexopoulos, N. Nikolakis, and G. Chryssolouris, "Digital twin-driven supervised machine learning for the development of artificial intelligence applications in manufacturing," *International Journal of Computer Integrated Manufacturing*, vol. 33, no. 5, pp. 429–439, 2020.

[7] B. Putz, M. Dietz, P. Empl, and G. Pernul, "Ethertwin: Blockchain-based secure digital twin information management," *Information Processing & Management*, vol. 58, no. 1, p. 102425, 2021.

[8] S. Lang, T. Reggelin, M. Müller, and A. Nahhas, “Open-source discrete-event simulation software for applications in production and logistics: An alternative to commercial tools?” *Procedia Computer Science*, vol. 180, pp. 978–987, 2021.

[9] D. Elia, S. Fiore, and G. Aloisio, “Towards hpc and big data analytics convergence: Design and experimental evaluation of a hpda framework for escience at scale,” *IEEE Access*, vol. 9, pp. 73 307–73 326, 2021.

[10] G. Chryssolouris, *Manufacturing Systems: Theory and Practice*, 2nd ed. Springer New York, NY, 2006.

[11] B. Vogel-Heuser and D. Hess, *Guest Editorial Industry 4.0–Prerequisites and Visions*, 13th ed. IEEE Transactions on Automation Science and Engineering, 2016.

[12] S. xin Zhou, *The Practical Applications of Industry 4.0 Technology to a New Plant for Both Manufacturing Technique and Manufacturing Process in New Product Introduction*, volume 8 ed. IEEE Access, 2021.

[13] L. Bassi, *Industry 4.0: Hope, hype or revolution?* IEEE 3rd International Forum on Research and Technologies for Society and Industry (RTSI)IEEE Access, 2017.

[14] P. Fantini, M. Pinzone, and M. Taisch, “Placing the operator at the centre of industry 4.0 design: Modelling and assessing human activities within cyber-physical systems,” *Computers & Industrial Engineering*, vol. 139, p. 105058, 2020.

[15] M. Lom, O. Pribyl, and M. Svitek, “Industry 4.0 as a part of smart cities,” in *2016 Smart Cities Symposium Prague (SCSP)*. IEEE, 2016, pp. 1–6.

[16] N. Bazmohammadi, A. Madary, J. C. Vasquez, H. B. Mohammadi, B. Khan, Y. Wu, and J. M. Guerrero, *Microgrid Digital Twins: Concepts, Applications, and Future Trends*, vol.10, 2284 - 2302 ed. IEEE Access, 2021.

[17] R. Avendano, F. Bontadini, N. Mulder, and D. Zaclicever, “Latin america’s faltering manufacturing competitiveness: What role for intermediate services?” 2020.

[18] U. ECLAC, “Economic survey of latin america and the caribbean 2021: Labour dynamics and employment policies for sustainable and inclusive recovery beyond the covid-19 crisis,” 2021.

[19] N. CELAC, “Innovation for development: the key to a transformative recovery in latin america and the caribbean. main messages,” 2021.

[20] J. Banks, J. S. CARSONII, L. Barry *et al.*, “Discrete-event system simulationfourth edition,” 2005.

[21] S. H. Choi and C. Choi, “A simulation budget allocation procedure for finding both extreme designs simultaneously in discrete-event simulation,” *IEEE Access*, vol. 8, pp. 93 978–93 986, 2020.

[22] S. R. Gonzalez, H. Jalali, and N. I. Van, “A multiobjective stochastic simulation optimization algorithm,” *European Journal of Operational Research*, vol. 284, no. 1, pp. 212–226, 2020.

[23] O. J. Shukla, G. Soni, A. Sujil, and R. Kumar, “Discrete event system framework for analysis and modeling of job shop scheduling system,” in *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2016, pp. 2420–2424.

[24] L. Fumagalli, E. Negri, E. Sottoriva, A. Polenghi, and M. Macchi, “A novel scheduling framework: Integrating genetic algorithms and discrete event simulation,” 2018.

[25] F. Della Croce, R. Tadei, and G. Volta, “A genetic algorithm for the job shop problem,” *Computers & Operations Research*, vol. 22, no. 1, pp. 15–24, 1995.

[26] A. Karakra, F. Fontanili, E. Lamine, J. Lamothe, and A. Taweel, “Pervasive computing integrated discrete event simulation for a hospital digital twin,” in *2018 IEEE/ACS 15th international conference on computer systems and Applications (AICCSA)*. IEEE, 2018, pp. 1–6.

[27] A. Fuller, Z. Fan, C. Day, and C. Barlow, “Digital twin: Enabling technologies, challenges and open research,” *IEEE access*, vol. 8, pp. 108 952–108 971, 2020.

[28] E. H. Bowman, "The schedule-sequencing problem," *Operations research*, vol. 7, no. 5, pp. 621–624, 1959.

[29] W. Y. Ku and J. C. Beck, "Mixed integer programming models for job shop scheduling: A computational analysis," *Computers and Operations Research*, vol. 73, 2016.

[30] K. R. Baker and D. Trietsch, *Principles of sequencing and scheduling.* John Wiley & Sons, 2013.

[31] H. Izadkhah, "P, np, np-complete, and np-hard problems," in *Problems on Algorithms.* Springer, 2022, pp. 497–511.

[32] M. R. Garey and D. S. Johnson, "Computers and intractability," *A Guide to the*, 1979.

[33] K. Tadaki, T. Yamakami, and J. C. Lin, "Theory of one-tape linear-time turing machines," *Theoretical Computer Science*, vol. 411, no. 1, pp. 22–43, 2010.

[34] A. Manne, *On the Job-Shop Scheduling Problem*, vol. 8, issue 2, 219-223 ed. Operations Research, 1960.

[35] E. Kondili and R. Sargent, *A general algorithm for scheduling batch operations.* Department of Chemical Engineering, Imperial College, 1988.

[36] S. M. Lee, J. Kim, M. Kim, H. Kim, and N. J. Choi, "Industrial hpc activities in korea," in *2011 International Conference on High Performance Computing & Simulation.* IEEE, 2011, pp. 814–818.

[37] T. Sterling, M. Brodowicz, and M. Anderson, *High performance computing: modern systems and practices.* Morgan Kaufmann, 2017.

[38] M. Mubarak, C. D. Carothers, R. B. Ross, and P. Carns, "Enabling parallel simulation of large-scale hpc network systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 1, pp. 87–100, 2016.

[39] B. Acun, N. Jain, A. Bhatele, M. Mubarak, C. D. Carothers, and L. V. Kale, "Preliminary evaluation of a parallel trace replay tool for hpc network simulations," in *European Conference on Parallel Processing.* Springer, 2015, pp. 417–429.

[40] G. Chapuis, S. Eidenbenz, N. Santhi, and E. J. Park, "Simian integrated framework for parallel discrete event simulation on gpus," in *2015 Winter Simulation Conference (WSC).* IEEE, 2015, pp. 1127–1138.

[41] G. Gurrala, A. Dimitrovski, S. Pannala, S. Simunovic, and M. Starke, "Parareal in time for fast power system dynamic simulations," *IEEE Transactions on Power Systems*, vol. 31, no. 3, pp. 1820–1830, 2015.

[42] R. Diao, S. Jin, F. Howell, Z. Huang, L. Wang, D. Wu, and Y. Chen, "On parallelizing single dynamic simulation using hpc techniques and apis of commercial software," *IEEE Transactions on Power Systems*, vol. 32, no. 3, pp. 2225–2233, 2016.

[43] E. Strohmaier, J. J. Dongarra, H. W. Meuer, and H. D. Simon, "Recent trends in the marketplace of high performance computing," *Parallel Computing*, vol. 31, no. 3-4, pp. 261–273, 2005.

[44] L. Yilmaz, W. Chan, I. Moon, T. Roeder, C. Macal, and M. Rossetti, "Simian integrated framework for parallel discrete event simulation on gpus."

[45] J. Leng, D. Wang, W. Shen, X. Li, Q. Liu, and X. Chen, "Digital twins-based smart manufacturing system design in industry 4.0: A review," *Journal of manufacturing systems*, vol. 60, pp. 119–137, 2021.

[46] C. Zhang, Y. Chen, H. Chen, and D. Chong, "Industry 4.0 and its implementation: A review," *Information Systems Frontiers*, pp. 1–11, 2021.

[47] L. Zhang, L. Zhou, L. Ren, and Y. Laili, "Modeling and simulation in intelligent manufacturing," *Computers in Industry*, vol. 112, p. 103123, 2019.

[48] B.-h. Li, B.-c. Hou, W.-t. Yu, X.-b. Lu, and C.-w. Yang, "Applications of artificial intelligence in intelligent manufacturing: a review," *Frontiers of Information Technology & Electronic Engineering*, vol. 18, no. 1, pp. 86–96, 2017.

[49] X. Li, H. Liu, W. Wang, Y. Zheng, H. Lv, and Z. Lv, "Big data analysis of the internet of things in the digital twins of smart city based on deep learning," *Future Generation Computer Systems*, vol. 128, pp. 167–177, 2022.

[50] H. R. Hasan, K. Salah, R. Jayaraman, M. Omar, I. Yaqoob, S. Pesic, T. Taylor, and D. Boscovic, "A blockchain-based approach for the creation of digital twins," *IEEE Access*, vol. 8, pp. 34 113–34 126, 2020.

[51] Y. Wu, K. Zhang, and Y. Zhang, "Digital twin networks: A survey," *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 13 789–13 804, 2021.

[52] R. Sahal, S. H. Alsamhi, K. N. Brown, D. O'Shea, C. McCarthy, and M. Guizani, "Blockchain-empowered digital twins collaboration: smart transportation use case," *Machines*, vol. 9, no. 9, p. 193, 2021.

[53] M. Faheem, A. Murphy, and C. Reaño, "Gpu-accelerated discrete event simulations: Towards industry 4.0 manufacturing," in *2021 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2021, pp. 1–7.

[54] N. Kumbhare, A. Marathe, A. Akoglu, H. J. Siegel, G. Abdulla, and S. Hariri, "A value-oriented job scheduling approach for power-constrained and oversubscribed hpc systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 6, pp. 1419–1433, 2020.

[55] M. Weiland, A. Jackson, N. Johnson, and M. Parsons, "Exploiting the performance benefits of storage class memory for hpc and hpda workflows," *Supercomputing Frontiers and Innovations*, vol. 5, no. 1, pp. 79–94, 2018.

[56] G. Da Col and E. Teppan, "Google vs ibm: A constraint solving challenge on the job-shop scheduling problem," *arXiv preprint arXiv:1909.08247*, 2019.

[57] A. Munshi, "The opencl specification," in *2009 IEEE Hot Chips 21 Symposium (HCS)*. IEEE, 2009, pp. 1–314.

[58] H. Fisher, "Probabilistic learning combinations of local job-shop scheduling rules," *Industrial scheduling*, pp. 225–251, 1963.

[59] T. Kloud and F. Koblasa, "Solving job shop scheduling with the computer simulation," *The International Journal of Transport & Logistics*, vol. 3, pp. 7–17, 2011.

[60] J. A. Gromicho, J. van Hoorn, G. Timmer *et al.*, “Exponentially better than brute force: solving the jobshop scheduling problem optimally by dynamic programming,” Tech. Rep., 2009.

Appendices

.1 Appendix 1.

DES software/ system	*Characteristics*
Tecnomatix Plant Simulation	Plant Simulation is an object-oriented simulation and optimization software for manufacturing and logistics. It can be used to optimize facility layouts, evaluate efficiency of facilities, analyze bottlenecks and balance of the production line, etc. It supports real-time communication with Computer-Aided Design (CAD), Computer-Aided Production Engineering (CAPE), Enterprise Resource Planning (ERP), and databases.
DELMIA/Quest	DELMIA is a product of Dassault Systèmes, it includes several modules: (1) DEP for manufacturing process; (2) QUEST for logistics analysis; (3) DPM for assembly process; (4) HUMAN for human factors and ergonomics; (5) ROBOTICS for robot simulation; and (6) VNC for virtual numerical control machine simulation. Quest is a simulation software mainly targeting modelling, experiments, analyses of facility layouts and flexibility of production processes.
AnyLogic	AnyLogic is widely used modelling and simulation software in many different fields, such as logistics, supply chain, pedestrian traffic simulation, manufacturing, urban planning, Petri net, geographic information system (GIS), spread of disease, etc. It is the first simulation software using Unified Modeling Language (UML), and supports both discrete event and continuous simulation.

FlexSim	FlexSim is an object-oriented software with a comprehensive modelling library. The modelling speed can be significantly increased with highly customizable objects. In addition to the flexibility, it also provides industry-leading 3D animation and user-interface.
Lanner Witness	Witness is one of the predominant simulation software and provides an interactive modelling environment with a comprehensive production modelling library. In addition, features like 3D visualization and virtual reality (VR) support provides more applications to users.
Arena	Unlike other commercial simulation software, Arena uses an entity-based, flowcharting methodology for modelling dynamic processes. Entities in an Arena model proceed through a flow chart of the process and seize control of resource capacity as they are processed.

Table 1: Commercial DES software. Source [1]

.2 Appendix 2.

Machine0:	job_0_task_1	job_3_task_1	job_2_task_3	job_5_task_3	job_1_task_4	job_4_task_4
	[1,4]	[13,18]	[18,27]	[28,38]	[38,48]	[48,51]
Machine1:	job_1_task_0	job_3_task_0	job_5_task_0	job_0_task_2	job_4_task_1	job_2_task_4
	[0,8]	[8,13]	[13,16]	[16,22]	[22,25]	[27,28]
Machine2:	job_0_task_0	job_2_task_0	job_1_task_1	job_4_task_0	job_3_task_2	job_5_task_5
	[0,1]	[1,6]	[8,13]	[13,22]	[22,27]	[49,50]
Machine3:	job_2_task_1	job_5_task_1	job_0_task_3	job_3_task_3	job_1_task_5	job_4_task_5
	[6,10]	[16,19]	[22,29]	[29,32]	[48,52]	[52,53]
Machine4:	job_1_task_2	job_4_task_2	job_2_task_5	job_3_task_4	job_5_task_4	job_0_task_5
	[13,23]	[25,30]	[30,37]	[37,45]	[45,49]	**[49,55]**
Machine5:	job_2_task_2	job_5_task_2	job_1_task_3	job_0_task_4	job_4_task_3	job_3_task_5
	[10,18]	[19,28]	[28,38]	[38,41]	[41,45]	[45,54]

Table 2: Results using Google OR-Tool for optimization. Data set ft06. Long version.

Machine0:	job_0_task_1	job_2_task_3	job_3_task_1	job_5_task_3	job_1_task_4	job_4_task_4
	0,1,1,4	2,3,18,27	3,1,27,32	5,3,37,47	1,4,47,57	4,4,64,67
Machine1:	job_1_task_0	job_0_task_2	job_3_task_0	job_4_task_1	job_5_task_0	job_2_task_4
	1,0,0,8	0,2,8,14	3,0,14,19	4,1,19,22	5,0,22,25	2,4,27,28
Machine2:	job_0_task_0	job_2_task_0	job_4_task_0	job_1_task_1	job_3_task_2	job_5_task_5
	0,0,0,1	2,0,1,6	4,0,6,15	1,1,15,20	3,2,32,37	5,5,60,61
Machine3:	job_2_task_1	job_0_task_3	job_5_task_1	job_3_task_3	job_1_task_5	job_4_task_5
	2,1,6,10	0,3,14,21	5,1,25,28	3,3,37,40	1,5,57,61	**4,5,67,68**
Machine4:	job_1_task_2	job_0_task_5	job_2_task_5	job_3_task_4	job_4_task_2	job_5_task_4
	1,2,20,30	0,5,30,36	2,5,36,43	3,4,43,51	4,2,51,56	5,4,56,60
Machine5:	job_2_task_2	job_0_task_4	job_5_task_2	job_1_task_3	job_3_task_5	job_4_task_3
	2,2,10,18	0,4,21,24	5,2,28,37	1,3,37,47	3,5,51,60	4,3,60,64

Table 3: Results using Discrete Event Simulation development model. Data set ft06.Long version.

Machine0:	job_0_task_0	job_1_task_0	job_8_task_0	job_6_task_1	job_3_task_2	job_4_task_1	job_9_task_1	job_7_task_1	job_5_task_6	job_2_task_1
	[0,29]	[29,72]	[72,148]	[148,185]	[185,256]	[256,262]	[262,275]	[275,361]	[361,408]	[408,493]
Machine1:	job_3_task_0	job_5_task_1	job_6_task_0	job_9_task_0	job_8_task_1	job_4_task_2	job_2_task_0	job_7_task_2	job_0_task_1	job_1_task_5
	[0,81]	[84,86]	[86,132]	[132,217]	[217,286]	[286,308]	[308,399]	[399,445]	[445,523]	[637,665]
Machine2:	job_5_task_0	job_3_task_1	job_4_task_0	job_7_task_0	job_1_task_1	job_6_task_3	job_9_task_2	job_8_task_4	job_0_task_2	job_2_task_3
	[0,84]	[84,179]	[179,193]	[193,224]	[224,314]	[314,327]	[327,388]	[421,506]	[523,532]	[532,606]
Machine3:	job_5_task_3	job_6_task_2	job_8_task_2	job_4_task_4	job_2_task_2	job_0_task_3	job_1_task_4	job_3_task_7	job_9_task_7	job_7_task_9
	[138,233]	[233,294]	[294,370]	[370,396]	[493,532]	[532,568]	[568,637]	[637,735]	[735,787]	[813,892]
Machine4:	job_3_task_3	job_1_task_2	job_4_task_5	job_5_task_8	job_7_task_4	job_0_task_4	job_8_task_8	job_6_task_9	job_9_task_8	job_2_task_9
	[256,355]	[355,430]	[430,499]	[499,505]	[519,551]	[568,617]	[668,694]	[698,753]	[787,877]	[887,920]
Machine5:	job_5_task_2	job_4_task_3	job_8_task_3	job_6_task_5	job_7_task_3	job_0_task_5	job_9_task_6	job_2_task_5	job_1_task_7	job_3_task_9
	[86,138]	[308,369]	[370,421]	[421,442]	[445,519]	[617,628]	[628,675]	[699,709]	[753,799]	[799,842]
Machine6:	job_6_task_4	job_3_task_4	job_9_task_3	job_5_task_7	job_8_task_6	job_7_task_5	job_0_task_6	job_1_task_6	job_2_task_7	job_4_task_9
	[327,359]	[359,368]	[388,395]	[408,473]	[517,557]	[557,645]	[645,707]	[707,753]	[753,842]	[842,895]
Machine7:	job_3_task_6	job_5_task_9	job_4_task_7	job_8_task_7	job_6_task_8	job_2_task_6	job_0_task_7	job_7_task_8	job_1_task_8	job_9_task_9
	[420,505]	[505,530]	[530,579]	[579,668]	[668,698]	[709,721]	[721,777]	[777,813]	[813,885]	**[885,930]**
Machine8:	job_5_task_4	job_3_task_5	job_9_task_4	job_4_task_6	job_6_task_7	job_2_task_4	job_7_task_6	job_8_task_9	job_0_task_8	job_1_task_9
	[233,281]	[368,420]	[420,484]	[499,520]	[520,609]	[609,699]	[699,718]	[718,792]	[792,836]	[885,915]
Machine9:	job_5_task_5	job_1_task_3	job_6_task_6	job_8_task_5	job_9_task_5	job_4_task_8	job_7_task_7	job_3_task_8	job_2_task_8	job_0_task_9
	[281,353]	[430,441]	[442,474]	[506,517]	[517,593]	[593,665]	[718,766]	[766,788]	[842,887]	[887,908]

Table 4: Results using Google OR-Tool for optimization. Data set ft10 Thomson and Fisher. Long version.

Machine0:	job_0_task_0	job_1_task_0	job_4_task_1	job_8_task_0	job_2_task_1	job_7_task_1	job_6_task_1	job_3_task_2	job_9_task_1	job_5_task_6
	0, 0, 0, 29	1, 0, 29, 72	4, 1, 72, 78	8, 0, 78, 154	2, 1, 154, 239	7, 1, 239, 325	6, 1, 325, 362	3, 2, 362, 433	9, 1, 548, 561	5, 6, 799, 846
Machine1:	job_2_task_0	job_0_task_1	job_3_task_0	job_4_task_2	job_5_task_1	job_6_task_0	job_8_task_1	job_1_task_5	job_7_task_2	job_9_task_0
	2, 0, 0, 91	0, 1, 91, 169	3, 0, 169, 250	4, 2, 250, 272	5, 1, 272, 274	6, 0, 274, 320	8, 1, 320, 389	1, 5, 389, 417	7, 2, 417, 463	9, 0, 463, 548
Machine2:	job_4_task_0	job_5_task_0	job_1_task_1	job_0_task_2	job_7_task_0	job_3_task_1	job_2_task_3	job_6_task_3	job_9_task_2	job_8_task_4
	4, 0, 0, 14	5, 0, 14, 98	1, 1, 98, 188	0, 2, 188, 197	7, 0, 197, 228	3, 1, 250, 345	2, 3, 345, 419	6, 3, 434, 447	9, 2, 561, 622	8, 4, 665, 750
Machine3:	job_0_task_3	job_2_task_2	job_1_task_4	job_4_task_4	job_6_task_2	job_5_task_3	job_8_task_2	job_3_task_7	job_7_task_9	job_9_task_7
	0, 3, 197, 233	2, 2, 239, 278	1, 4, 278, 347	4, 4, 347, 373	6, 2, 373, 434	5, 3, 434, 529	8, 2, 529, 605	3, 7, 690, 788	7, 9, 978, 1057	9, 7, 1075, 1127
Machine4:	job_1_task_2	job_0_task_4	job_4_task_5	job_3_task_3	job_7_task_4	job_2_task_9	job_6_task_9	job_5_task_8	job_8_task_8	job_9_task_8
	1, 2, 188, 263	0, 4, 263, 312	4, 5, 373, 442	3, 3, 442, 541	7, 4, 614, 646	2, 9, 727, 760	6, 9, 823, 878	5, 8, 911, 917	8, 8, 1093, 1119	9, 8, 1127, 1217
Machine5:	job_4_task_3	job_0_task_5	job_5_task_2	job_1_task_7	job_2_task_5	job_6_task_5	job_7_task_3	job_8_task_3	job_3_task_9	job_9_task_6
	4, 3, 272, 333	0, 5, 333, 344	5, 2, 344, 396	1, 7, 463, 509	2, 5, 509, 519	6, 5, 519, 540	7, 3, 540, 614	8, 3, 614, 665	3, 9, 821, 864	9, 6, 1028, 1075
Machine6:	job_0_task_6	job_1_task_6	job_6_task_4	job_3_task_4	job_2_task_7	job_7_task_5	job_9_task_3	job_5_task_7	job_4_task_9	job_8_task_6
	0, 6, 344, 406	1, 6, 417, 463	6, 4, 463, 495	3, 4, 541, 550	2, 7, 593, 682	7, 5, 682, 770	9, 3, 770, 777	5, 7, 846, 911	4, 9, 911, 964	8, 6, 964, 1004
Machine7:	job_0_task_7	job_1_task_8	job_2_task_6	job_3_task_6	job_4_task_7	job_6_task_8	job_5_task_9	job_7_task_8	job_8_task_7	job_9_task_9
	0, 7, 406, 462	1, 8, 509, 581	2, 6, 581, 593	3, 6, 605, 690	4, 7, 690, 739	6, 8, 793, 823	5, 9, 917, 942	7, 8, 942, 978	8, 7, 1004, 1093	**9, 9, 1217, 1262**
Machine8:	job_2_task_4	job_0_task_8	job_3_task_5	job_1_task_9	job_4_task_6	job_5_task_4	job_6_task_7	job_7_task_6	job_9_task_4	job_8_task_9
	2, 4, 419, 509	0, 8, 509, 553	3, 5, 553, 605	1, 9, 605, 635	4, 6, 635, 656	5, 4, 656, 704	6, 7, 704, 793	7, 6, 793, 812	9, 4, 812, 876	8, 9, 1119, 1193
Machine9:	job_1_task_3	job_6_task_6	job_0_task_9	job_2_task_8	job_5_task_5	job_3_task_8	job_4_task_8	job_7_task_7	job_8_task_5	job_9_task_5
	1, 3, 263, 274	6, 6, 540, 572	0, 9, 572, 593	2, 8, 682, 727	5, 5, 727, 799	3, 8, 799, 821	4, 8, 821, 893	7, 7, 893, 941	8, 5, 941, 952	9, 5, 952, 1028

Table 5: Results using Discrete Event Simulation development model ft10 Data set. Long version.

Machine0:	job_16_task_0	job_4_task_2	job_14_task_0	job_8_task_0	job_15_task_1	job_18_task_2	job_17_task_0	job_11_task_1	job_9_task_2	job_12_task_0
	[0,37]	[37,63]	[63,69]	[69,145]	[145,240]	[240,251]	[251,337]	[337,348]	[349,413]	[413,498]
Machine1:	job_15_task_0	job_4_task_1	job_16_task_2	job_18_task_0	job_14_task_1	job_9_task_0	job_10_task_0	job_17_task_1	job_11_task_2	job_5_task_1
	[0,2]	[15,37]	[50,71]	[71,140]	[140,201]	[203,288]	[288,366]	[366,440]	[450,478]	[478,530]
Machine2:	job_4_task_0	job_16_task_1	job_13_task_0	job_18_task_1	job_11_task_0	job_9_task_1	job_7_task_0	job_5_task_0	job_15_task_4	job_14_task_3
	[1,15]	[37,50]	[50,145]	[145,196]	[196,286]	[288,349]	[349,380]	[380,464]	[464,489]	[489,538]
Machine3:	job_4_task_3	job_16_task_3	job_8_task_1	job_15_task_2	job_18_task_3	job_10_task_1	job_9_task_3	job_11_task_3	job_14_task_4	job_0_task_3
	[63,84]	[84,173]	[173,249]	[249,321]	[321,410]	[410,446]	[446,493]	[493,539]	[539,592]	[592,654]
Machine4:	job_4_task_4	job_16_task_4	job_14_task_2	job_15_task_3	job_18_task_4	job_17_task_2	job_9_task_4	job_0_task_4	job_5_task_2	job_11_task_4
	[84,156]	[173,228]	[235,304]	[342,407]	[410,484]	[486,574]	[574,664]	[664,708]	[708,756]	[758,788]

Machine0:	job_0_task_0	job_19_task_0	job_13_task_1	job_2_task_1	job_6_task_1	job_1_task_0	job_10_task_2	job_7_task_2	job_5_task_3	job_3_task_1
	[498,527]	[527,540]	[540,639]	[639,678]	[684,745]	[745,788]	[788,799]	[799,831]	[855,902]	[944,1015]
Machine1:	job_0_task_1	job_19_task_1	job_2_task_0	job_6_task_0	job_13_task_2	job_7_task_1	job_1_task_1	job_3_task_0	job_12_task_2	job_8_task_3
	[530,539]	[540,547]	[547,638]	[638,684]	[684,736]	[740,786]	[788,863]	[863,944]	[944,954]	[993,1033]
Machine2:	job_0_task_2	job_17_task_3	job_19_task_2	job_2_task_2	job_6_task_2	job_12_task_1	job_8_task_2	job_1_task_3	job_3_task_3	job_10_task_4
	[539,588]	[588,636]	[636,712]	[712,802]	[802,834]	[834,908]	[908,993]	[1009,1055]	[1055,1140]	[1140,1161]
Machine3:	job_17_task_4	job_19_task_3	job_13_task_3	job_6_task_3	job_5_task_4	job_7_task_3	job_1_task_2	job_12_task_3	job_2_task_4	job_3_task_4
	[654,733]	[733,785]	[785,883]	[883,915]	[915,921]	[921,940]	[940,1009]	[1009,1098]	[1098,1143]	**[1143,1165]**
Machine4:	job_19_task_4	job_2_task_3	job_10_task_3	job_13_task_4	job_7_task_4	job_6_task_4	job_3_task_2	job_8_task_4	job_1_task_4	job_12_task_4
	[788,833]	[833,845]	[845,901]	[901,944]	[944,980]	[980,1010]	[1019,1028]	[1033,1059]	[1059,1131]	[1131,1164]

Table 6: Results using Google OR-Tool for optimization. Data set ft20 Thomson and Fisher. Long version.

Machine0:	job_0_task_0	job_1_task_0	job_8_task_0	job_2_task_1	job_12_task_0	job_3_task_1	job_4_task_2	job_11_task_1	job_5_task_3	job_6_task_1	job_7_task_2	job_14_task_0	job_16_task_0	job_17_task_0	job_10_task_2	job_19_task_0	job_9_task_2	job_15_task_1	job_13_task_1	job_18_task_2
	0,0,0,29	1,0,29,72	8,0,72,148	2,1,148,187	12,0,187,272	3,1,272,343	4,2,343,369	11,1,369,380	5,3,380,427	6,1,427,488	7,2,488,520	14,0,520,526	16,0,526,563	17,0,563,649	10,2,649,660	19,0,660,673	9,2,673,737	15,1,737,832	13,1,857,343	18,2,970,981
Machine1:	job_2_task_0	job_0_task_1	job_1_task_1	job_3_task_0	job_4_task_1	job_5_task_1	job_6_task_0	job_7_task_1	job_9_task_0	job_10_task_0	job_11_task_2	job_8_task_3	job_14_task_1	job_15_task_0	job_17_task_1	job_12_task_2	job_18_task_0	job_19_task_1	job_16_task_2	job_13_task_2
	2,0,0,91	0,1,91,100	1,1,100,175	3,0,175,256	4,1,256,278	5,1,278,330	6,0,330,376	7,1,376,422	9,0,422,507	10,0,507,585	11,2,585,613	8,3,613,653	14,1,653,714	15,0,714,716	17,1,716,790	12,2,790,800	18,0,800,869	19,1,869,876	16,2,919,330	13,2,956,1008
Machine2:	job_4_task_0	job_5_task_0	job_7_task_0	job_0_task_2	job_11_task_0	job_2_task_2	job_1_task_3	job_3_task_3	job_6_task_2	job_8_task_2	job_9_task_1	job_12_task_1	job_10_task_4	job_13_task_0	job_14_task_3	job_16_task_1	job_18_task_1	job_19_task_2	job_17_task_3	job_15_task_4
	4,0,0,14	5,0,14,98	7,0,98,129	0,2,129,178	11,0,178,268	2,2,268,358	1,3,358,404	3,3,404,489	6,2,489,521	8,2,521,606	9,1,606,667	12,1,667,741	10,4,741,762	13,0,762,857	14,3,857,906	16,1,906,919	18,1,919,970	19,2,970,1046	17,3,1046,358	15,4,1110,1135
Machine3:	job_8_task_1	job_0_task_3	job_1_task_2	job_4_task_3	job_2_task_4	job_5_task_4	job_3_task_4	job_7_task_3	job_6_task_3	job_10_task_1	job_11_task_3	job_9_task_3	job_12_task_3	job_15_task_2	job_14_task_4	job_13_task_3	job_16_task_3	job_17_task_4	job_18_task_3	job_19_task_3
	8,1,148,224	0,3,224,286	1,2,286,355	4,3,369,390	2,4,390,435	5,4,435,441	3,4,489,511	7,3,520,539	6,3,539,571	10,1,585,621	11,3,621,667	9,3,737,784	12,3,800,889	15,2,889,961	14,4,961,1014	13,3,1014,1112	16,3,1112,1201	17,4,1201,1280	18,3,1280,441	19,3,1369,1421
Machine4:	job_0_task_4	job_5_task_2	job_2_task_3	job_3_task_2	job_4_task_4	job_1_task_4	job_7_task_4	job_6_task_4	job_8_task_4	job_10_task_3	job_11_task_4	job_14_task_2	job_9_task_4	job_12_task_4	job_17_task_2	job_15_task_3	job_13_task_4	job_16_task_4	job_18_task_4	job_19_task_4
	0,4,286,330	5,2,330,378	2,3,378,390	3,2,390,399	4,4,399,471	1,4,471,543	7,4,543,579	6,4,579,609	8,4,653,679	10,3,679,735	11,4,735,765	14,2,765,834	9,4,834,924	12,4,924,957	17,2,957,1045	15,3,1045,1110	13,4,1112,1155	16,4,1201,1256	18,4,1369,543	**19,4,1443,1488**

Table 7: Results using the DES implementation. Data set ft20 Thomson and Fisher. Long version.

.3 Appendix 3.

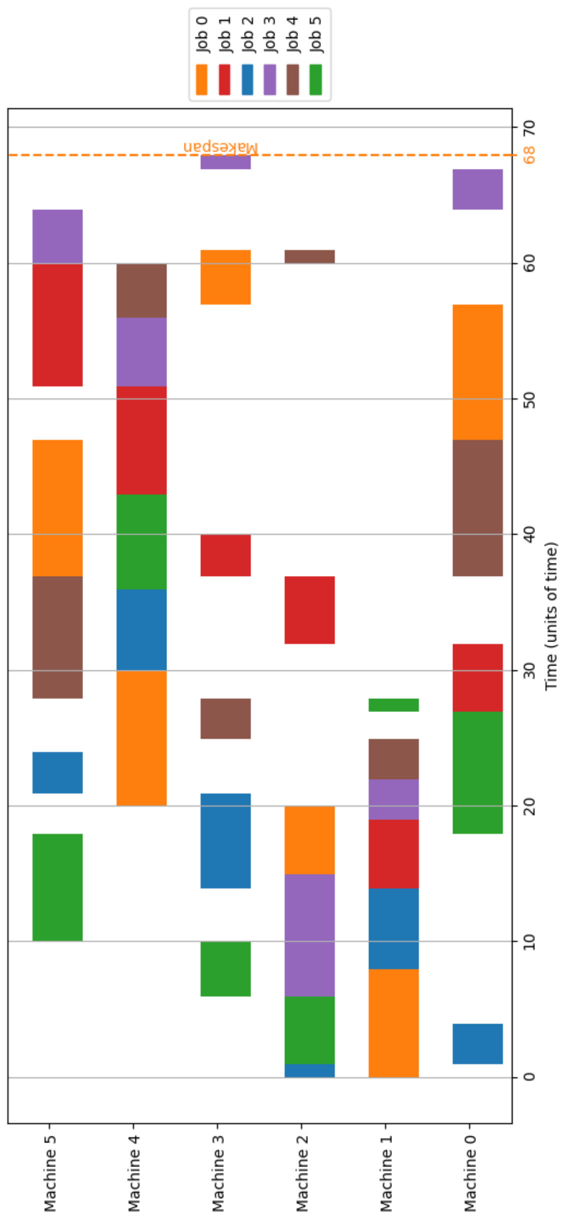


Figure 1: Serial Simulation with DES Model ft06 Thomson and Fisher Data-set. Long version.

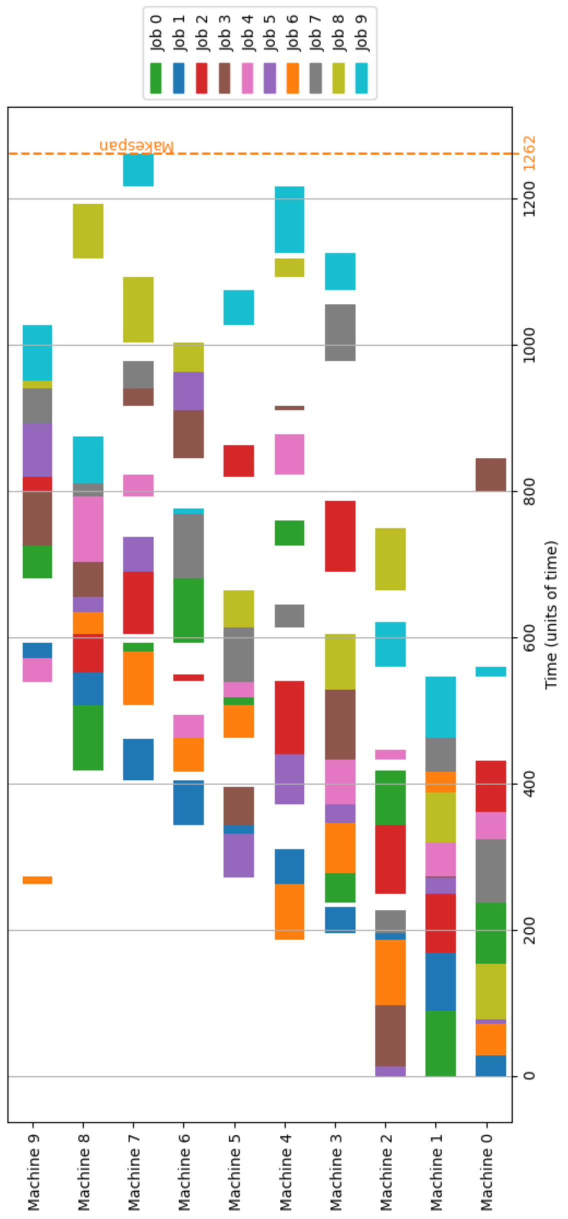


Figure 2: Serial Simulation with DES Model ft10 Thomson and Fisher Data-set. Long version.

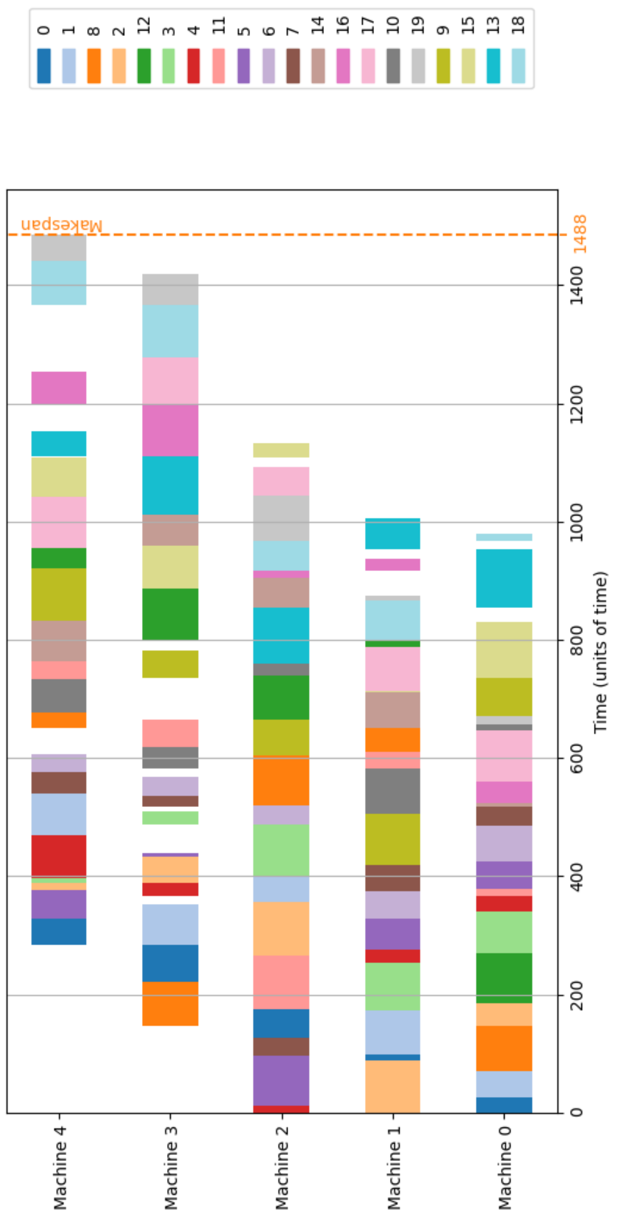


Figure 3: Serial Simulation with DES Model ft20 Thomson and Fisher Data-set. Long version.