Dual Resource Constrained Flexible Job Shop Problem using Hybrid Genetic Algorithm

Application to Quality Control Laboratory Scheduling

Diogo Miguel Afonso Marta

Thesis to obtain the Master of Science Degree in

Mechanical Engineering

Supervisors: Prof. João Miguel da Costa Sousa
Prof. Susana Margarida da Silva Vieira

Examination Committee

Chairperson: Prof. Carlos B. Cardeira
Supervisor: Prof. João Miguel da Costa Sousa
Member of the Committee: Prof. Carlos Augusto Santos Silva

December 2021

This work was created using LaTeX typesetting language
in the Overleaf environment (www.overleaf.com).

Acknowledgments

I would like to thank Eng. Bernardo Firme, for his support and guidance throughout these last months. Without you this project would not be possible. I would also like to show my appreciation to Eng. Tiago Coito and Eng. Miguel Martins for their initial guidance in this thesis. I would like to acknowledge my dissertation supervisor Professor João Sousa for his insight, support and sharing of knowledge that has made this Thesis possible.

I would also like to show my appreciation towards my Parents for their sacrifices and love through the years.

Last but not least, to all my friends and colleagues. A special thanks to Tomás, Rodrigo and Amaara. Thank you for all your help

To each and every one of you – Thank you.

Abstract

Quality control laboratory scheduling has huge potential to reduce costs and increase production. However, this is not widely discussed or supported by academics. This thesis proposes a hybrid genetic algorithm with and without variable neighborhood search to minimize the total completion time.
The problem is formulated as an extension of a Dual Resource Constrained Flexible Job Shop Problem for the allocation of both machine and worker resources. This extension, divides each job in three phases allowing the worker to perform other jobs in between these mandatory presential stages. While maintaining the traditional structure of genetic algorithms, this work presents a way of creating the Initial Population, based on increment tables which groups the most compatible jobs together. Also, rules regarding the allocation of resources to sequential tasks were introduced. A new combination of crossover operations with weighted probabilities are proposed, proving that combining multiple types of crossovers achieves fitter results. Additionally, a novel combination of mutation operations is implemented and a variable neighborhood search with four different structures was introduced alongside a dynamic termination criteria that adapts the parameters of the algorithm. A tuning of the parameters was performed and the final results were compared to the benchmark. This study is competitive with the benchmark for small instances, achieving the optimal solution for seven of them. For medium sized instances the proposed genetic algorithm surpasses the literature. For large sized instances this study generated significant results surpassing the compared results by 57%.

Keywords

Quality Control Laboratory; Dual Resource Constrained Scheduling; Flexible Job Shop Problem; Hybrid Genetic Algorithm; Total Completion Time.

Resumo

O escalonamento de um laboratório de controlo de qualidade tem um enorme potencial para reduzir custos e aumentar a produção. Todavia, este problema tem tido pouco suporte por parte da academia. Esta tese propõe um algoritmo genético híbrido com a possibilidade de realizar uma procura local para minimizar o tempo total de conclusão. Este problema é formulado como uma extensão do escalonamento flexível de duplo constrangimento de recursos para a alocação de tanto máquinas como trabalhadores. Esta extensão, divide cada trabalho em três fases, possibilitando ao analista ser alocado em mais do que uma tarefa ao mesmo tempo, em máquinas diferentes, durante os períodos em que este está livre.

Mantendo a estrutura típica de um algoritmo genético, o trabalho desenvolvido combina novas maneiras de criar a População Inicial, baseadas em tabelas incrementais que junta os trabalhos mais compatíveis. Ainda, regras para alocação sequencial de recursos foram introduzidas. Uma nova combinação de operações de crossover com probabilidades baseadas em pesos é proposta, provando que combinando diferentes tipos de crossover obtém-se melhores resultados. Adicionalmente, uma nova combinação de mutações é também aplicada e uma procura local com quatro estruturas distintas foi introduzida juntamente com um critério de paragem dinâmico que adapta os parâmetros do algoritmo. Um afinação dos parâmetros foi realizada e os resultados finais foram comparados com os da literatura. Este estudo é competitivo com a literatura para pequenas instâncias, alcançando soluções óptimas para sete destas. Para instâncias médias o algoritmo genético ultrapassa os resultados da literatura. Para grandes instâncias este estudo gerou resultados significantemente melhores ultrapassando os estudos

anteriores por 57%.

Palavras Chave

Laboratório de Controlo de Qualidade; Escalonamento Flexível de Duplo Constrangimento de Recursos; Problema de Job shop Flexível, Algoritmo Genético Híbrido; Tempo Total de Conclusão

Nomenclature

Indices

j	subscript for jobs
i	subscript for operations
k	subscript for machines
h	subscript for workers
s	subscript for tasks

Dimensions

n	number of jobs
q_j	number of operations of job j
m	number of machines
w	number of workers
N_{ij}	number of tasks in operation i of job j

Sets

J	set of jobs
K	set of machines
K	set of workers
O_j	set of operations of job j
K_{ij}	set of machines that can be used for operation O_{ij}
W_{ij}	set of workers that can be used for operation O_{ij}

Parameters

p_{ij}	processing time of operation i of job j
ρ^s_{ijs}	time after the start of processing of operation i of job j
ρ^d_{ijs}	duration of intervention s in operation i of job j
M	very large number

Variables

t_{ij}	start time of operation i of job j
x_{ijk}	binary assignment of operation i of job j to machine k
α_{ijh}	binary assignment of operation i of job j to worker h
$\beta_{iji'j'}$ and $\gamma iji'j'$	sequencing variables
$\mathcal{J}$	total completion time
c_j	completion time of job j

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

DRC	Dual Resource Constrained
DRCFJSP	Dual Resource Constrained Flexible Job Shop Problem
FJSP	Flexible Job Shop Problem
GA	Genetic Algorithm
JSP	Job Shop Problem
SA	Simulated Annealing
VNS	Variable Neighborhood Search
VDO	Vibration Damping Optimization

1

Introduction

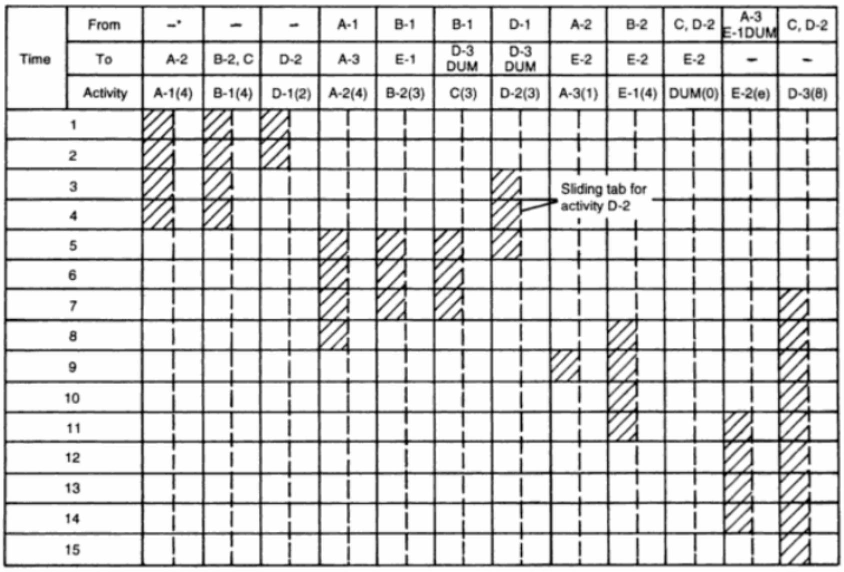
Contents

1.1 Motivation

Rome wasn't built in a day - or is it the case that Rome wasn't built without scheduling? For over thousands of years, humans have been using scheduling as a tool to build and coordinate operations. Scheduling is central to how jobs are planned and completed and provides order to businesses. Without scheduling, workers simply cannot work. In its infancy, scheduling was grasped as a concept, for example comprehending activities and sequencing of operations. This transitioned to executing wide-scale operations such as building pyramids (2780 BCE), to the great wall of China (7 BCE) which required scheduling hundreds of thousands of workers. Mosteiro dos Jerónimos is a masterpiece of construction of the 16th century which took over a century to plan and build. Convento de Mafra, dates to the 18th century and has over 1200 rooms. None of these monuments could ever have been accomplished without some form of scheduling.

However, most empirical evidence and sources suggest that formal scheduling processes were only implemented during the 19th Century. In today's scheduling tools, a commonly used chart to show the allocation of resources in time is the Gantt Chart, propagated by Henry Gantt during the early 20th century. The purpose of this tool was to help organisations plan repetitive tasks, measure productivity levels, and demonstrate how employees' resources can be allocated more efficiently. The Gantt Chart revolutionised the way people work and is now deployed as a powerful tool to help project managers schedule and plan projects. Therefore, utilising scheduling processes are instrumental in aiding efficiency and productivity in the workplace. [3]

Also, one of the earliest scheduling tools is Karol Adamiecki's 'Harmonygraph' (figure 1.1). Karol sought to create 'work harmonization' and demonstrated the significance of creating practical scheduling and it has been argued that companies who implemented Karol's methods experienced an increase in productivity of up to 400% [3].



*The first column or strip represents activity A-1, where (4) indicates the estimated time to perform this activity. The dash in the 'From' row indicates that activity A-1 has no predecessor activities, and the A-2 in the 'To' row indicates that it is a successor to A-1.

Figure 1.1: The 'Harmonygraph' by Karol Adamiecki's [1]

The Y axis shows the time scale, and X axis shows a list of tasks. The duration of the tasks is shown by a vertical sliding tab. This chart also depicts the activities the predecessors and successors participated in. Although this was not published until 1931, Adamiecki's harmonygraph inspired the CPM (Critical Path Method) and PERT (Project Evaluation and Review Technique) systems.

The evolution of the scheduling science allows systems to be more complex, with a higher number of activities and sequencing of operations. Scheduling is required for complex manufacturing systems. Manufacturing systems have evolved and become increasingly industrialized, more reliant on technology with state-of-the-art thinking and machinery. Since industrial manufacturing first started it has been gone through four pivotal developments. Each revolution is unique and serves as major turning points within society. Each revolution adopted new methods of manufacturing, which resulted in significant improvement to products and competitiveness of companies.

Dating back to the middle of the 18th century, the First Industrial Revolution relied on the transition to new manufacturing processes using water and steam. Industry 2.0, also known as "The Technological Revolution" introduced superior electrical technology. The Third Industrial Revolution (1970) began with the first computer era, where automated systems became a commonality, although still very dependent on human input and intervention. In the XX century companies targeted for large batches, low costs and standardization. [4]

Currently, society is experiencing the fourth wave of the Industrial revolution, Industry 4.0, and academia is already looking forward to a fifth. In the XXI century, standardization of the internet paved way for a fast-paced world and accelerated the expansion of globalisation. The internet enabled the interconnectedness of countries, goods, news, and people on an unprecedented scale. With easier access to news and trends, this led to rapidly changing consumer demands, and resulted in industries needing to become more adaptable to satisfy demand and allow for greater customization The era of mass production is gone, more and more industries focus on the concept of mass customization. Therefore, the integration of smart systems, machines embedded with sensors, software and other technologies with the purpose of connecting and exchanging data with other devices and systems over the internet is of major importance. With this technological advances, industries are capable of producing smaller batches with smaller inventories, different products at the same time and have a better control of the material flow in the supply chain while maintaining profitable margins. [5]

A particular industry that requires the adaptation and transition to the IoT and Industry 4.0 is the Pharmaceutical Industry. Pharmaceutical companies operate in one of the most competitive and regulated markets, where compliance with Good Manufacturing Practices (GMP) and Good Laboratory Practices (GLP) is mandatory to commercialize any drug. Without such tight regulations, medicines could become serious dangerous to public health [6]. Therefore, companies that successfully integrate in their production systems the IoT may obtain considerable better margins over the remaining market, despite the tight regulations and the abundance of competition.

On the other hand, the future of the pharmaceutical sector is predicted to be prosperous. For example, the race for the Covid-19 vaccine has stimulated innovation across the industry, but the sector has been building momentum for years. Prior to the current pandemic, between 2006 and 2020, the Industry's Research and Development (R&D) was growing at an annual rate of 7.6% in the USA [7]. These growing costs have been mainly associated with the shifting paradigm forcing the industry to change from a one-drug-fits-all approach to more targeted drugs for small and specific patient and therapeutic groups (mass customization) [4].

The challenge related to the high R&D spend is the rising expectations of investors for a high return of investment (ROI). Margins are shrinking which dissuades potential investors from investing in new medicines. R&D has not been living up to the expectations as the total number of molecular entities commercialized in past years did not match with the extraordinary high R&D costs [6].

On the other side, there is a growing expectation for pharmaceutical companies to have a higher proportion of developed products regulated and accepted by markets, and developed within faster time frames. This in turn leads to higher profit margin. With both arguments in mind, corporations begin to follow the principles of Industry 4.0 in order to apply them to the pharmaceutical industry which is known as Pharma 4.0. Pharma 4.0 has brought a series of challenges including the scheduling and rescheduling problem. The scheduling and rescheduling problem entails the search for the best possible set of decisions which allows for an increase in profits, by cutting costs in the production and quality control phases. This achieves higher throughput rates, lower energy consumption and more effective and planned maintenance, whilst maintaining high quality levels. [4].

During its development life cycle, a drug must be constantly monitored with laboratory tests to assess its quality. In this scenario, quality control laboratories serves as a critical function in pharmaceutical production and control [8]. It is responsible for ensuring that goods comply with predefined standards by following guidelines and practices. More specifically, it is responsible for monitoring manufacturing processes through the assessment of samples taken at different stages of the manufacturing process. These can be taken at the early stages to raw materials, at mid stages and also at the final stages when the finished product is developed. The assessment of these samples is completed by comparing with the standard pre-defined quality metrics. Quality control makes it possible to meet high product quality standards [9] by promptly detecting deviations in the quality of a manufacturing stream.

The role of managing jobs within QC laboratories is a complicated and complex task. For example, there are thousands of different tasks, which each require the delegation of specific skills and specialised instruments in order to execute each test. Laboratory management involves resources (both personal and equipment) planning and scheduling, analysis prioritization results evaluation and documentation.

The management of jobs within contract manufacturing organisations (manufacturers that produces goods under the brand of its clients), causes greater complications. This is because these manufacturers deal with a large array of projects and handle a variety of materials. Given the high mix of products and tests it is important to develop effective strategies for laboratory management. Laboratory information management systems (LIMS) used in pharmaceutical industry often lack on essential features such as schedule planning and stock management [6].

The purpose of this thesis is to propose a solution to automatically schedule real-sized QC laboratories. In this chapter, a literature review on scheduling methods will be presented mainly focusing on flexible and dual resource constrained job shop problems as it represents the problem at hand. At the end of the chapter a guideline of the thesis is given.

1.2 Scheduling in Manufacturing System

1.2.1 Flexible Job Shop Problem

Scheduling is a challenging problem in manufacturing shop floors and is known to be strongly an NP-hard problem. It can be defined as the allocation of finite resources over a period of time in order to optimize one or more objectives.

Shop-floors have limitations in the amount of different tasks a certain machine and worker can perform. Therefore, studies have included flexibility as an additional limitation to better simulate real manufacturing complex conditions. Brucker and Schlie (1991) [10] introduced a Job Shop Problem (JSP) with machines as the only flexibility parameter achieving reasonable results for instances smaller than 3 jobs.

Later, in 1997, Dauzère-Pérès and Paulli [11] formulated a Flexible Job Shop Problem (FJSP) in which an extended version of the disjunctive graph model was presented, allowing for no distinction between reassigning or resequencing an operation. This problem has been studied significantly by researchers utilising genetic algorithms [12], [13] .

1.2.2 Dual Resource Constrained Problem

The dual resource constraint formulation is a generalisation of the multi-resource constraint, and includes three sub-problems: (i) Assigning operations to resources of machines, (ii) Assigning operations to resources of workers and (iii) Sequencing the operations on the machines considering workers in order to optimize the performance measure [14]. This allows for simplifications and can be utilised for simpler computational algorithms and in turn decreases computational time for larger problems [2].

Most of the literature predominantly focuses on equipment as the sole limiting constraint within JSP. However, workers often appear as the bottleneck in many shop-floors, especially if both workers and machines reach capacity constraints. Kher [15] focused on evaluating the policies for deploying workers and the dispatching rules (”when” and ”where”) to offer a near-perfect delivery performance for vital customers with very rigid delivery guidelines in the Dual Resource Constrained (DRC) job shop environments.

1.2.3 Dual Resource Constrained Flexible Job Shop Problem

In 1997, Patel [16] proposed, in his master thesis, a combination of the two problems, a Scheduling of flexible Manufacturing Systems Under Dual-Resource Constraints using Genetic Algorithms. Six different dispatching rules were formulated and used with eight performance criteria. These were compared with single resource constrained JSP. The results demonstrated that depending on the number of resources being constrained, different dispatching rules proved better results. For the DRC problem, the shortest processing time performed the best. In 2000, ElMaraghy et al. [17] proposed an algorithm where machines were set in work-centers of a 2 by 2 formation. Also using six different dispatching rules, ElMaraghy et al., drew a similar conclusion to Patel's, except ElMaraghy et al demonstrated that the DRC did not prove better results for any dispatching rule in particular.

Labour flexibility is often expensive, as more training is required to be administered. In 2001 Felan and Fry [18] investigated the effect of having different levels of training across the workforce (Multi-level heterogeneous flexibility). Their results revealed that a combination of workers with very high flexibility and workers with no flexibility, performed higher than those with near equal flexibility.

Yue et al. [19] investigated cross-training policies in DRC parallel job shop. The workers were required to learn new skills, as additional parts were added to the system. Moreover, if workers stopped performing a certain job for a significant period of time, a forgetting model would be implemented to simulate the time they would take to perform it once they were assigned to that job again. This curve reflected on the experience they have gathered and the time they had stopped doing the given work. In 2011, Xianzhou and Zhenhe [20], presented a new immune Genetic Algorithm which proved to have a higher level of convergence precision when solving a DRC flexible job shop problem, through merging immune and genetic algorithms.

Other meta-heuristic algorithms have proved to yield higher quality results in the Dual Resource Constrained Flexible Job Shop Problem (DRCFJSP) while minimizing the makespan. Lei and Guo (2014) [21] proposed a Variable Neighborhood Search (VNS) composed of two neighbourhood search procedures and a restarting mechanism. In the same year Yazdani et al. [22] presented a Simulated Annealing (SA) and a Vibration Damping Optimization (VDO) algorithms with the same minimization criteria.

Apart from the makespan minimization, Paksi and Ma'ruf (2015) [23] developed a GA algorithm utilising indirect chromosome representation with two layers in order to reduce delays. In 2017 Zhang et al. [24] proposed a Particle Swarm Optimization with a three layered encoding scheme in order to reduce the production period and cost. Zhong et al. (2018) [25] presented an algorithm that could lower the makespan and total processing cost. They stated that the local search algorithms, tabu search, classical metaheuristic methods, or binary particle swarm optimization could not achieve the same results. A branch population genetic algorithm based on compressed time-window scheduling strategy was implemented, which did not hamper the quality of the initial population. An improvement in the makespan was achieved by 7%.

1.2.4 Human Factors and Uncertainty

Researchers have extensively explored whether human factors should be considered in optimization algorithms, i.e. methodologies which include factors such as fatigue and productivity. Despite the reasonable considerations in favour of the human factors, Helander in 2000 [26] presented seven reasons for not considering human factor in the production system development process which, among others, included the unpredictability of human behaviour. This hardly quantifiable factor leads to uncertainty in the production times which often complicates planning.

Scheduling with uncertain production times is also know as fuzzy scheduling. Lang and Li [27] explore the uncertain operation time constraints using grey simulation technology and Non-dominated Sorting Genetic Algorithm II considering delivery satisfaction, cost, energy consumption and noise pollution as the optimized objects. In 2016 Gao et al. [28] presented an artificial bee colony algorithm considering fuzzy processing time and new job insertion which demanded new rescheduling operations. This operation of performing new assignments based on newly arrived information is often called dynamic scheduling and it is another branch of scheduling problems.

1.2.5 A wider overview into Scheduling problems

This dissertation cannot possibly cover all different types of manufacturing scheduling. Nevertheless, there are some that, due to its similarities or relevant considerations to the problem at hand, inspired the development of this thesis and therefore, are worth to be mentioned.

Unpredictable events such as failure of a machine, sickness of a worker or new urgent batches arriving at the shop floor, happen on any industry. This forces operations to be rescheduled, also known as dynamic scheduling. In 2010, Araz and Salum [29] proposed a real-time (dynamic) scheduling approach to select a predetermined scheduling rule in DRC manufacturing systems. The model requires a combination of Artificial Neural Networks, fuzzy inference system and simulation to provide the knowledge base. Cunha (2017) [4], proposed a dynamic scheduling master thesis in which a flexible dynamic environment existed. To cope with the dynamic environment two ant inspired multi-agent algorithms were proposed. Recently in 2020, Andrade-Pineda et al. [30] proposed a solution for an automobile collision repair shop where the re-scheduling needs, like due-date changes delay in arrival, changes in job processing time and rush jobs were common. Reasonable schedules could be achieved under five seconds.

In order to minimize the makespan, Zheng and Wang (2016) [31] developed a knowledge-guided fruit fly optimization algorithm (KGFOA) which is an improvement from the simpler fruit fly optimization (FOA) which enabled guiding the search process. In 2018, Guo et al. [32] proposed a hybrid genetic algorithm which hybridizes genetic algorithm (GA) with variable neighborhood search (VNS) to overcome GA's slow convergence speed due to it's unguided mutation.

Some studies also included loading and unloading times of fixtures, considering the influence of resource requirement similarity among different operations, as well as the time to shift workers between assignments. Wu et al. [33], used similarity-based scheduling algorithm for setup-time reduction (SSA4STR) and later introduced an improved non-dominated sorting genetic algorithm II (NSGA-II) to optimize the DRFJSP when loading and unloading fixtures, also called, DRFJSP-LU. A goal in this research was to minimize the total time fixtures were mounted and dismounted on/off the machines, i.e. tasks using the same fixtures were encouraged to be completed sequentially, as it could save time loading and unloading the fixture.

When more than a single objective is desirable to be optimized at the same time, often the problem is called a multi-objective optimization. In 2017, Gong et al. [34] proposed an extension to the traditional genetic algorithm, a memetic algorithm with the objective to minimize the maximum completion time, the maximum workload of machines and the total workload of all machines. Approaching human factor indicators, green production and processing time simultaneously, Gong et al. (2018) [35] proposed a hybrid genetic algorithm to solve the problem. In 2019 Yazdani and Zandieh [36] proposed two types of multi-objective evolutionary algorithm including fast elitist non-dominated sorting genetic algorithm and non-dominated ranking genetic algorithm. The objective was to solve Multi-Objective DRCFJSP by minimizing the makespan and the critical machine workload simultaneously.

Some researchers have even drawn their attention to other constrained problems rather than exclusively focusing on single and dual resource constraints. For example Gao and Pan (2016) [37] proposed a multi-resource constrained flexible job shop scheduling problem by using a shuffled multi-swarm micro-migrating birds optimizer. The constraints were labor, maintenance equipment, tooling and machinery. This study was the first reported application of micro-evolutionary algorithms to solve flexible job shop problems. The experimental results and statistical analyses demonstrated that the presented SM-MBO algorithm clearly surpassed all of the other compared algorithms by a substantial margin. The authors also pointed that their simple solution representation (only machine assignment and operation sequence presented in the encoding), which was capable of decreasing the search space, was also a factor to be taken into consideration in the algorithm's success.

1.2.6 Laboratory Scheduling - Quality control specific applications

The scholars previously mentioned, haven't included the context of the Quality control laboratory scheduling. Whereby, workers can work on several jobs simultaneously, as it is only necessary to be present in 3 different time instances on the assigned machine for a given job, representing the setup, intermediate and disassembly/data processing tasks. Each and every study previously mentioned, did not allow workers to leave a certain machine unattended while it was performing a job.

In 2019 Cunha et al. [38] formulated a Mixed integer linear programming (MILP) algorithm to solve the DRCFJSP in the Quality control laboratory scheduling (QCLS) environment in order to minimize the makespan. The obtained solution could only reach the optimum for small instances where the number of jobs considered was equal or less than three.

Recently, in 2020 Akbar and Irohara [39] proposed seven metaheuristic algorithms with six different decoding schemes to solve what they named a multi-task simultaneous supervision dual resource-constrained scheduling problem. The multi-tasking refers to the ability of a machine to work independently of the worker and allows for parallel scheduling after an initial set up and a final supervision, much like this thesis problem, (without the sampling operation in middle). Although, it does not allowing for different skill levels in workers and machines (flexibility). The metaheuristics that proved the best results were the modified Permutation-based Genetic Algorithm and the Modified Bees Algorithm. The same authors, Akbar and Irohara [40], two year prior, presented a study explaining that this type of multi-tasking ability of workers could yield workload imbalances between operators. Social sustainability was therefore acquainted in a multiple objective model to minimize makespan and workload unbalances.

In 2021, Martins et al. [2], sought to investigate the QC scheduling problem, formulated a three-level dynamic heuristic with both a branch and cut and a tabu-search algorithm implemented for comparison. The results showed that the heuristic outperforms the other algorithms for large-sized instances. The present thesis results will be compared with the ones from Martins et al. [2] research.

1.2.7 Literature Review Scheme

In this section, a table review (1.1 and 1.2) of the different scheduling methods approached in the literature is presented in chronological order based on the criteria proposed by Dhiflaoui et al. (2018) [14].

The structure of the approach is composed of four criteria:

(1) Are the Machines Flexible? (Can machines perform more than one task but are limited in the amount of tasks they can perform? ”yes” or ”no”).

(2) Are Workers Flexible? (Can Workers perform more than one task but are limited in the amount of tasks they can perform? ”yes” or ”no”).

(3) The optimization criteria (Makespan, Cost, Lateness, among others).

(4) The implemented approaches (Mixed integer programming, Genetic algorithm, Tabu search, among others).

1.3 Contributions

The added-value proposal can be summarized in the following topics:

Evaluation and development of previous literature work

The proposed framework is the first study conducted with a meta-heuristic algorithm in scheduling problems for the Quality Control laboratory environment. Therefore, the first hybrid Genetic Algorithm with Variable Neighborhood Search to solve such a problem is proposed and compared to bench mark results.

Adaptation to the Quality Control Scheduling Problem

Apart from operating outside the common literature field, the present work includes six novel implementations: (1) - A probabilistic choice in the Initial Population based on weights which correspond to different paths when selecting tasks. It proved to be best when kept on a random selection (section 5.3.1).

(2) - A Prohibition condition which controls the resources allocation which proved to improve results by 4% (section 5.3.2).

Table 1.1: Literature Review - part 1

Date	Author(s)	Machines Flexible?	Workers Flexible?	Optimization Criteria	Approach
1991	Brucker and Schlie	yes	no	Makespan	2D - Network
1997	Patel	yes	yes	Makespan	Genetic
1997	Dauzère-Pérès and Paulli	yes	no	Makespan	Tabu Search
2000	Eimaraghy et al.	yes	yes	Makespan	Genetic
2000	Kher	no	no	Percentage of tardy jobs and Root mean squared tardiness	ANOVA
2001	Felan and Fry	no	yes	Labour cost vs Labour perform given the flexibility	ANOVA
2007	Pezzellaa et al.	yes	no	Makespan	Genetic
2008	Yue et al.	no	yes	Mean flow time of jobs	ANOVA
2010	Araz and Salum	no	no	Multiple Criteria	Artificial neural networks and Fuzzy inference system
2011	Xianzhou and Zhenhe	yes	yes	Malespan	Genetic + Immune algorithm
2011	Lang and Li	yes	yes	Delivery satisfaction, Cost, Energy consumpiton and Noise pollution	Grey simulation technology and Non-dominated Sorting Genetic Algorithm- II (NSGA)-II
2014	Driss et al.	yes	no	Makespan	New Genetic Algorithm (NGA)
2014	Lei and Guo	yes	yes	Makespan	Variable Neighbourhood Search (VNS)
2015	Yazdani et al.	yes	yes	Makespan	Simulated annealing and Vibration Damping Optimization
2016	Paksi and Ma'ruf	yes	yes	Total Tardiness	Genetic
2016	Zheng and Whang	yes	yes	Makespan	Knowledge guided Fuit Fly Optimization
2016	Gao and Pan	yes	yes	Makespan	Shuffled Multi-swarm Micro-migrating Birds Optimization (SM 2 - MBO)

Table 1.2: Literature Review - Part 2

Date	Author(s)	Machines Flexible?	Workers Flexible?	Optimization Criteria	Approach
2016	Gao et al.	yes	yes	Minimize maximum fuzzy completion time	Artificial Bee Colony (ABC)
2017	Cunha	yes	yes	Optimizing unforseen events	Genetic and Variable Neighbourhood Search (VNS)
2017	Gong et al.	yes	yes	Maximum completion time, Maximum workload and Total workload	Memetic Algorithm
2017	Zhang et al.	yes	yes	Makespan and Cost	Hybrid discrete particle swarm optimization
2018	Gong et al.	yes	yes	Maximum Completion Time, maximum total worker cost, green-production related indicator	Newly Hybrid Genetic Algorithm (NHGA)
2018	Akbar and Irohara	no	no	Social Sustainability, Workload smoothness index, Makespan	NSGA-II
2018	Guo et al.	yes	yes	Makespan	Genetic and Variable Neighbourhood Search (VNS)
2018	Zhong et al.	no	yes	Makespan and Cost	Branch Population Genetic
2019	Cunha et al.	yes	yes	Makespan	MILP
2019	Yazdani et al.	yes	yes	Makespan, Critical machine workload and Total workload of machines simultaneously.	Fast elitist non-dominated sorting genetic algorithm (NSGA-II) and non-dominated ranking genetic algorithm (NRGA)
2020	Akbar and Irohara,	no	no	Makespan	Tabu Search, Simulated Annealing, Particle Swarm Optimization, Bees Algorithm, Artificial Bee Colony, and Grey wolf optimization.
2020	Andrade-Pineda et al.	yes	yes	Makespan and Due-date related criteria	Iterated greedy constructive heuristic and MILP
2021	Wu et al.	yes	yes	Makespan and Total Setup time	Similarity-based scheduling algorithm (SSA4STR) and non-dominated sorting genetic algorithm (NSGA-II)
2021	Martins et al.	yes	yes	Total completion time	Three-level dynamic heuristic

(3) - An Increment Initialization that combines compatible tests together which improves results by 5.3% (section 5.3.3).

(4) - A probabilistic choice in the Crossover operation (with the same logic as the Initial Solution) which improves the results when compared to performing Crossover operations with only one type of crossover. Improvements from a single point crossover are 5.8%, from a double point crossover are 4.8% and from an MPX crossover are 1.1% (section 5.4).

(5) - A Variable Neighborhood Search is implemented to improve GA's slow convergence speed which combines four different structures. This implementation proved fitter results for small instances (0.1% better) when compared to a GA without VNS (section 5.5).

(6) - A Dynamic Termination Criteria is proposed to allow the algorithm to adapt based on the number of non-improving iterations (section 5.6).

1.4 Organization of the Document

The remainder of this document is structured as follows:

Chapter 2 - The Dual Resource Constrained Flexible Job Shop Problem

This chapter presents the description of the problem, it's assumptions and it's mathematical formulation.

Chapter 3 - Genetic Algorithm

In this chapter the genetic algorithm is introduced and it's principles explained covering the five main phases of a GA: Initial Population (3.1), Fitness evaluation (3.2), Selection (3.3), Crossover (3.4), and Mutation (3.5). Also different encoding schemes presented in literature are described (3.1.1).

Chapter 4 - Proposed Hybrid Genetic Algorithm

Chapter 4 details the proposed Hybrid Genetic Algorithm explaining the logic behind each specific implementation (4.1 to 4.8). The variable neighborhood search is introduced and explained (4.9) as well as the Dynamic Termination Criteria (4.10).

Chapter 5 - Results

This chapter presents the description of the experiments, explaining the reason behind the different test instances utilized (5.1). Also, the studies conducted to tune the algorithm are presented (5.2 to 5.6) and a comparison between the final results this Hybrid GA produced and bench mark results is shown (5.7).

Chapter 6 - Conclusion

At last, in chapter 6 the conclusions for the present dissertation are drawn and possible future work is discussed.

2

The Dual Resource Constrained Flexible Job Shop Problem

Contents

2.1 Application to the Quality Control Laboratory Scheduling

As mentioned in section 1, the classical flexible job shop scheduling problem (FJSSP) solely treats machines as the only resource constraint, failing to consider the importance of the dominant role of qualified labour in production and manufacturing. Quality control laboratories in pharmaceutical industry requires both resources to be scheduled and often have a flexible workforce and machinery capable of performing different tasks.

The quality control laboratories scheduling problem can be summarised as follows: assign each operation (test) of each job (sample) to a machine (analytical instrument) and worker (analyst/chemist), to minimise an objective function. Each operation can only be processed by predetermined subsets of machines and workers. The machine is required for the full processing time of the test while the worker is only required at a number of predetermined intervals (tasks) during the processing time of the operation. These tasks are the machine setup, the preparation of sample and the materials and data processing at the end. All of these tasks must be performed by the same worker. In contrast to the literature reviewed, in this Quality Control Laboratory, workers can switch between tests, if they are present when they are required by the operations they are allocated to. Additionally, all tasks in an operation must be carried out by the same worker.

Some assumption are put forward:

- Each Machine can only process one operation at a time on any job.
- Each Worker may operate in more than one machine at a time, given that the worker respects the times he needs to be present at the machines he is assigned to.
- Each operation can be performed only once on one machine and its sequence is respected for every job.
- The operations of different jobs do not have precedence constraints, only the ones in the same job.
- A temporary interruption of an operation is not allowed after it has started.
- An operation of any job cannot be processed until its preceding operations are completed.
- The processing time corresponding to each operation are given in advance and are the same regardless the machine or worker that performs it.

2.2 Formulation of the Scheduling Problem

The following mathematical formulation was performed by [2]: an instance of the QC labs scheduling problem consists of a set of jobs, $J = \{J_1, ..., J_n\}$ with n being the total number of jobs. The set of machines is represented by: $K = \{K_1, ..., K_m\}$, the set of workers by $W = \{W_1, ..., W_w\}$ and the set of operations by $O_j = \{O_{1j}, ..., O_{qj}\}$ for each job j. The number of operations in job j is q_j. K_{ij} and W_{ij} are, respectively, the subsets of machines and workers that can process operation i of job j. Additionally, each operation can be divided by N_{ij} individual tasks, s that requires a worker present, either partially or fully. Each operation is characterised by its processing time p_{ij}, the start time, ρ^s_{ijs}, and the duration time, ρ^d_{ijs}. The machine processing time is always greater than the sum of the duration of the worker tasks and is the same as the processing time p_{ij}.

The main decision variables are the start time of operation i of job j, t_{ij}, the binary assignment of operations to machines, $x_{ijh} \in \{0, 1\}$ and the binary assignment of operations to workers, $\alpha_{ijh} \in \{0, 1\}$. Additionally, the following sequencing variables are used: $\beta_{iji'j'} \in \{0, 1\}$ is equal to 1 if O_{ij} is scheduled before $O_{i'j'}$, it is 0 otherwise; and $\gamma_{ijsi'j'} \in \{0, 1\}$ is equal to 1 if the task (worker intervention) s of O_{ij} is scheduled before the task s', it is 0 otherwise.

2.2.1 Objective Functions

The Objective function of the problem is to minimise the total completion time J (2.1) where c_j is the completion time of job j, i.e., the time required to complete the operations of that job.

$$\min \mathcal{J} = \sum_{j \in J} c_j, \quad \forall j \in J \tag{2.1}$$

2.2.2 Constraints

The completion time of each job, c_j must be greater or equal than the completion time of the last operation of that job as expressed below in:

$$c_j \geq t_{c_j j} + p_{c_j j} \tag{2.2}$$

Each test can only be assigned to one suitable machine (2.3) and worker (2.4):

$$\sum_{k \in K} x_{ijk} = 1, \quad \forall j \in J, i \in O_j, k \in K_{ij} \tag{2.3}$$

$$\sum_{h \in W} \alpha_{ijk} = 1, \quad \forall j \in J, i \in O_j, h \in W_{ij} \tag{2.4}$$

Constraint (2.5) guarantees the precedence is respected between operations of the same job.

$$t_{ij} \geq t_{(i-1)j} + p_{(i-1)j} \tag{2.5}$$

For the sequencing of operations in machines, constraint (2.6) ensures that the start time of any O_{ij} is greater or equal than the finish time of any $O_{i'j'}$ that is scheduled before ($\beta_{iji'j'} = 0$) in the same machine ($x_{ijk} = x_{i'j'k}$).

$$t_{ij} \geq t_{i'j'} + p_{i'j'} - (2 - x_{ijk} - x_{i'j'k'} + \beta_{iji'j'})M, \quad \forall j \in J,\, i \in O_j,\, O_{ij} \neq O_{i'j'},\, k \in K_{ij} \cap K_{i'j'} \tag{2.6}$$

As the prior constraint is disjunctive, the big M formulation is adopted, Similarly, constraint (2.7) guarantees that any O_{ij} scheduled before any other $O_{i'j'}$ in the same machine ($\beta_{iji'j'} = 1$ and $x_{ijk} = x_{i'j'k}$) is finished before the later starts.

$$t_{i'j'} \geq t_{ij} + p_{ij} - (3 - x_{ijk} - x_{i'j'k'} - \beta_{iji'j'})M, \quad \forall j \in J,\, i \in O_j,\, O_{ij} \neq O_{i'j'},\, k \in K_{ij} \cap K_{i'j'} \tag{2.7}$$

Two constraints are required to bound for $\beta_{iji'j'} = 0$ and $\beta_{iji'j'} = 1$. In the same way, constraints (2.8) and (2.9) ensure that a worker cannot perform overlapping tasks.

$$t_{i'j'} + \rho^s_{ijs} \geq t_{i'j'} + \rho^s_{i'j's'} + \rho^d_{i'j's'} - (2 - \alpha_{ijh} - \alpha_{i'j'h} + \gamma ijsi'j's')M, \quad \forall j \in J,\, i \in O_j,\, s \in N_{ij},\, O_{ij} \neq O_{i'j'} \tag{2.8}$$

$$t_{i'j'} + \rho^s_{i'j's'} \geq t_{ij} + \rho^s_{ijs} + \rho^d_{ijs} - (3 - \alpha_{ijh} - \alpha_{i'j'h} - \gamma ijsi'j's')M, \quad \forall j \in J,\, i \in O_j,\, s \in N_{ij},\, O_{ij} \neq O_{i'j'} \tag{2.9}$$

3

Genetic Algorithm

Contents

A Genetic Algorithm (GA) is a meta-heuristic search algorithm inspired by Charles Darwin's theory of natural evolution. This algorithm emulates the process of natural selection where the fittest individuals (with favourable genetics) have higher chances to propagate their genes to the next generation. The fittest individuals have a set of adaptive traits in nature, these advantageous differences can appear in size, shape/form and intelligence. These adaptive traits increase the creature's chance of survival in a certain environment, thus, making the individual superior and a more sought-after candidate for reproduction purposes. In contrast, the genetically inferior individuals are likely to disappear from the gene pool altogether.

For each offspring that is born, besides the inherited genes from the parents, offspring also generate random mutations in their genes which can lead to better adaptation and consequentially higher probabilities of gene propagation. With this natural mechanism species are able to keep the strongest genes in the gene pool, therefore, increasing the species chances to prosper whilst adapting to an ever changing environment.

It is worth mentioning that the artificial GA, while replicating the main concepts of natural evolution has some relevant differences. First and foremost, an initial solution has to be created. Therefore, a spontaneous and often random generation of the individuals is performed. Secondly, the individuals do not age in the artificial algorithm. They are simply discarded or kept in the population depending on their fitness value. This means that the fittest individuals may exist in numerous generations until the termination criteria is met and the algorithm stops.

Bearing these similarities and differences in mind, in an artificial Genetic Algorithm there are five main phases to be considered:

1) - Initial Population creation

2) - Fitness Evaluation of the population

3) - Selection

4) - Crossover

5) - Mutation

The following sections from 3.1 to 3.5 explain the main principles behind each phase:

3.1 Initial Population

This process begins with a set of generated individuals which together form the Population. Each individual is a solution to the problem and is generally randomly generated in accordance to the problem restrictions. Individuals are characterized by a set of parameters also known as Genes. A set of predefined number of genes constitutes a chromosome (solution) which in turn is the same as an individual. This correct ordering of genes forms a chromosome which is also known as the process of encoding.

The following figure 3.1 illustrates this concepts. In blue, the population, which consists of four individuals/chromosomes/solutions represented in green. Each chromosome has six genes, represented in red.

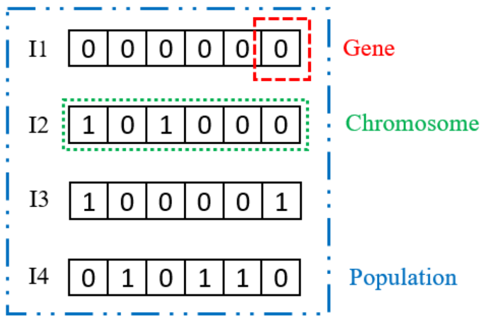


Figure 3.1: Population, Chromosomes and Genes definition

To form the Initial Population each solution needs to be correctly encoded. This can be done in various ways and depending on the problem, one path may be better adapted to it. The following section 3.1.1 presents the four main types of encoding schemes found in the literature for scheduling problems.

3.1.1 Encoding schemes

In scheduling problems, each solution is often represented by a code that must contain an enough amount of information to enable it to be converted into a scheduling solution when the decoding process is later applied. If this code has less information than it should, the feasible solution space narrows. On the other hand, if the code contains unnecessary information, the solution becomes more complex and may delay or complicate the entire process.

Mainly, the Encoding scheme in the literature is divided by 4 major representations which does not necessarily translate into 4 different decoding mechanisms since these, can vary from problem to problem in order to better accommodate to it's peculiarities.

(1) In this encoding method, each gene is represented by a quadruple string *(i,j,k,l)*, one for each operation, where ***i*** signifies the job which an operation belongs to; ***j*** characterizes the progressive number of that operation within job ***i***; ***k*** indicates the machine assigned to that operation and ***l*** indicates the worker assigned [21] [36]. An example of this representation follows:

Table 3.1: Decoding Methods - quadruple string

(1,1,4,2)	(2,1,2,2)	(4,1,1,1)	(4,2,5,1)	(2,2,1,2)	(3,1,2,2)	(1,2,3,1)	(1,3,1,1)	(3,2,1,2)	(3,3,2,2)

(2) A single chromosome is used by Akbar and Irohara [39] to encode one scheduling solution. The code contains information about the job sequence when evaluated in the decoding scheme where a

repetition of the same number in the encoding table represents the next operation of that job. In the particular case of table 3.2, the 8^{th} gene represents job 1 operation 2. This representation is also known as the operation-based encoding method. [33]

Table 3.2: Single Chromosome encoding

1	2	4	4	2	3	1	1	3	3

To perform the decoding, in this case, the attributed worker and machine may change every time a change in the operations order is done. For this representation [39] proposed 6 different decoding schemes and studied their effectiveness. The differences of this processes were dependent on the procedure used to select the machine and or operator i.e., whether the first available machine should be selected or the first available operator, or even the pair of machine and operation combined that could start the earliest or finish first.

(3) On a third type of encoding, Zheng and Wang (2016) [31] proposed two vectors, an operation sequence vector (OSV) and a resource assignment vector (RAV). OSV is used to denote a sequence of all the operations of all the jobs. RAV is used to denote an assignment of machines and workers. An example of this representation is seen below 3.3:

Table 3.3: Two Vector encoding

OSV:	$O_{1,1}$	$O_{2,1}$	$O_{4,1}$	$O_{4,2}$	$O_{2,2}$	$O_{3,1}$	$O_{1,2}$	$O_{1,3}$	$O_{3,2}$	$O_{3,3}$
RAV:	(M_4,W_2)	(M_2,W_2)	(M_1,W_1)	(M_5,W_1)	(M_1,W_2)	(M_2,W_2)	(M_3,W_1)	(M_1,W_1)	(M_1,W_2)	(M_2,W_2)

(4) Finally, the fourth type of encoding mechanism is a three layered chromosome (table 3.4) adding from the encoding type three a decomposition of the RAV vector into a Machine Assignment vector (MAS) and a Worker assignment vector (WAS). [34]

Table 3.4: Three Layered Chromosome

OSV:	1	2	4	4	2	3	1	1	3	3
MAS:	4	2	1	5	1	2	3	1	1	2
WAS:	2	2	1	1	2	2	1	1	2	2

3.2 Fitness Evaluation

The Fitness Evaluation quantifies all individuals in terms of the chosen objective function. A fitness score is assigned to each solution, according to the chosen objective function, which will correlate with the probability that the individual has of being selected for reproduction and thus remaining or not in the gene pool. In order to allow the evaluation of a solution an individual needs to be correctly encoded with the genes (encoding process). After the encoding is completed it is then possible to decompose the encoding scheme (decoding), in accordance with the problem constraints, to allow for the fitness evaluation. The following scheme 3.2 illustrates this concepts:

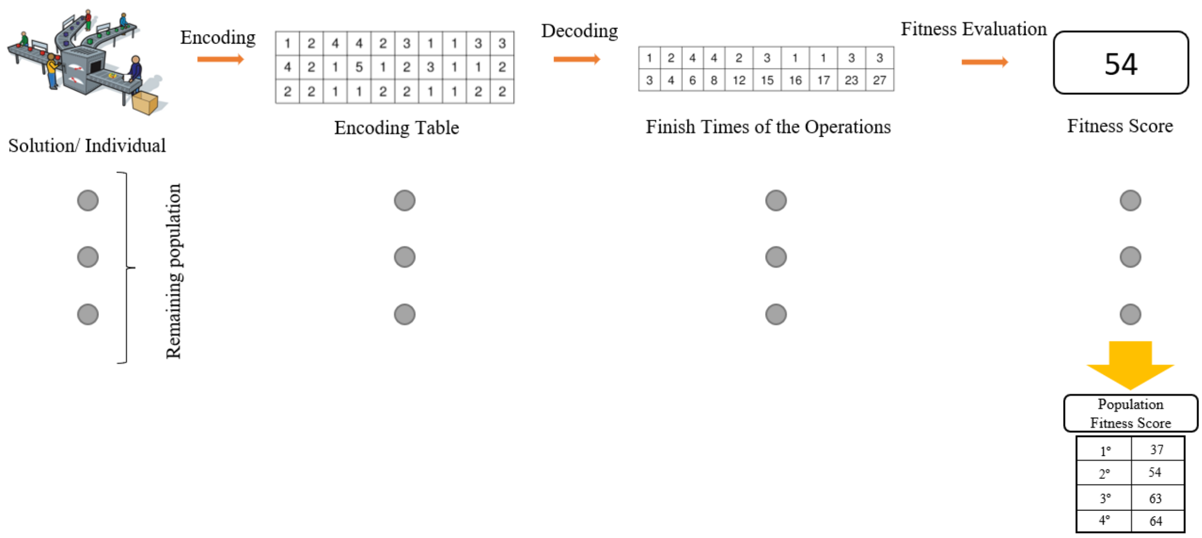


Figure 3.2: Fitness Evaluation Process

In figure 3.2 a certain solution is encoded (on a three layer encoding scheme) which then follows a process of decoding, which respects the equations defined in chapter 2, resulting in the Finish Times of each operations. With this information, based on a certain objective function, the fitness score of this individual is calculated and compared to the rest individuals in the same population. The represented individual is the second best in the population with a total fitness score of 54 time instances.

3.3 Selection

The selection phase is about choosing which population will be able to produce offspring. As it happens in the natural world, the fittest individuals have higher chances of passing on their genes. Often in artificial algorithms, the selection is performed by tournament or roulette wheel selection. In 2005, Zhang et al. [41] compared both methods and concluded that tournament selection is more efficient due to its higher convergence speed, which is important for large problems (which is the case). Therefore, in the present work, the former was chosen. Also, due to the tournament selection the best solutions might be discarded (as they might not be chosen for a tournament) and therefore an elitism operation is implemented to ensure the continuation of the best genes of the population.

3.4 Crossover

Crossover operations are about selecting which genes from which parents are passed to the off-springs (2 parents will produce 2 offspring). Often Crossover techniques rely on selecting random points in the parents chromosomes and swapping the genes they correspond to in order to build the off-springs encoding. The most common in literature are the single-point and the double-point crossover which will be covered in the next chapter.

3.5 Mutation

Mutation in an artificial genetic algorithm is the process of randomly selecting one or more genes from an offspring and then changing these genes while maintaining the solution feasibility.

After all of these steps the standard genetic algorithm repeats until a termination criteria is reached.

4

Proposed Hybrid Genetic Algorithm

Contents

The novel contribution of this thesis arises from using a Hybrid genetic algorithm to solve the DRC flexible job shop problem in a Quality Control Laboratory Environment which, to the author's best knowledge, has never been done before. The implemented Hybrid Genetic Algorithm contains the traditional phases in (3.1 to 3.5) with additional problem specific implementations. A flow chart of the general implementation of the hybrid GA is presented in figure 4.1.

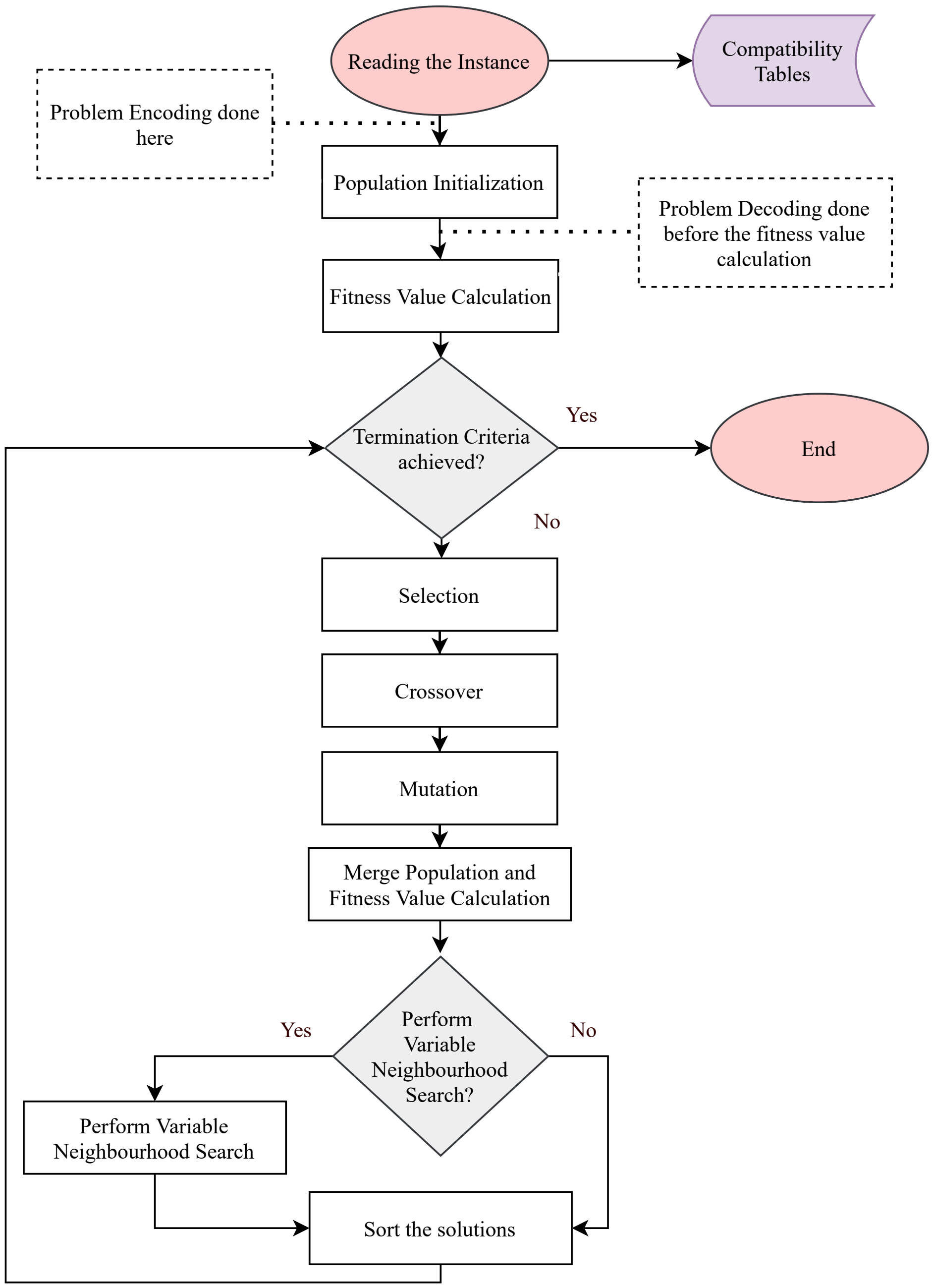


Figure 4.1: Implemented GA - flowchart

At first sight the implemented hybrid GA resembles the traditional one. Although, the differences to a standard GA do not lie in the sequencing of the steps, but in the steps itself. In this chapter these different phases will be explained in the following sections 4.1 to 4.10. It is worth mentioning that the Variable Neighbourhood Search is either chosen at the beginning of the algorithm or not, it does not depend on any probabilities.

4.1 Reading the Instance

In this phase, apart from the instance loading, two binary tables, the compatibility tables are produced. This idea was based on Xiao et al. (2021) [33] which applied this concept to the loading and unloading of fixtures in a scheduling problem and is now applied to the Quality Control Laboratory Scheduling environment. The tables have zeros (meaning can't perform) or ones (meaning can perform) which inform which jobs can be performed by which machines and workers. It is worth remarking that the Jobs appear with decimal numbers which mean the task of the current job to be performed. This was the notation chosen for the encoding table as it will be explained in section 4.3 Considering an example with 4 Jobs, 2 Workers and 7 machines the compatibility tables could be:

Table 4.1: Machines Compatibility Table

Job	M_1	M_2	M_3	M_4	M_5	M_6	M_7
1.1	1	0	0	0	1	0	0
2.1	0	0	1	1	1	0	0
3.1	1	1	1	0	0	0	1
3.2	0	0	0	1	0	0	0
4.1	0	1	0	1	0	0	1

Table 4.2: Analysts Compatibility Table

Job	**Worker 1**	**Worker 2**
1.1	1	0
2.1	1	0
3.1	0	1
3.2	0	1
4.1	1	1

This implementation is particularly useful to easily access information to ascertain whether a resource is able to perform a certain task or not. With tables 4.1 and 4.2 one can notice, for example, that job 1.1 is able to be performed by machine 1, machine 5 and worker 1.

4.2 Initial Population

In the Initial Population phase two distinct implementations which rely on different logic were created: the Initial Population Job Randomization (IPJR) and the Incremental Initial Population Worker Randomization (IIPWR). In both cases, a novel limitation in allocating resources was also implemented, namely a Prohibition Condition (PC) which restricted both workers and machines to work more than X and Y times in a row (respectively). The rationale behind the PC concept is that while assigning workers and machines to a certain job, both analysts and appliances may repeat themselves infinitely, as long as they can perform the chosen job, leading to inferior solutions solutions.

4.2.1 Initial Population with Job Selection

The logic behind this implementation is to produce an Initial Population in the most intuitive way possible. This is accomplished by firstly choosing an available job, and then allocating a random machine and worker that can perform the selected task.

In order to first select each job, a novel Job Selection Procedure is utilized,which combined three different methods: a Random Initialization, a Longest Processing Time selection and a Most Number Of Tests selection. All three of these methods can only choose from the pool of available tests previously defined. Random Initialization is as it's name suggests, each job is selected randomly. The Longest Processing Time selects the one with the longest processing time. The Most Number Of Tests chooses a job with the most amount of tests needing to be performed. In the latter two selections, if there are multiple jobs with the same conditions one of those is then chosen by randomization. In order to choose which implementation is used, weighs (w_1, w_2, w_3) are assigned to each alternative corresponding to the probability that each selection method has to be chosen.

In the figure 4.2, the Initial Population Job Randomization implementation can be seen:

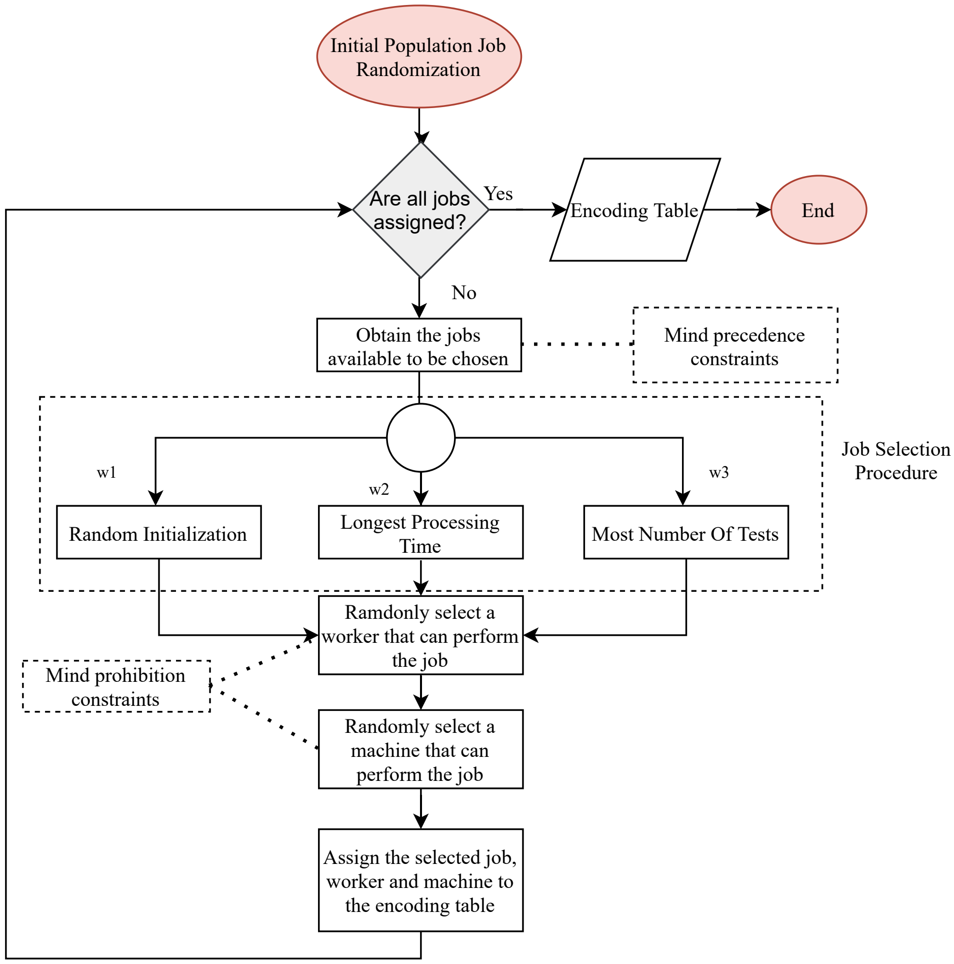


Figure 4.2: Initial Population with Job Selection flowchart

4.2.2 Incremental Initial Population with Worker Selection

The logic behind this implementation is based on the concept of puzzle pieces where, ideally, there is a sequence of jobs that can be perfectly grouped together, much like a puzzle. In the context of this problem, once a worker is assigned to a job and a machine, the worker is automatically assigned to three time instances on that machine that he must respect. The time in between these time instances is called Idle Gap and the problem is best solved when this interval of time is minimized. This means that if one could find a series of jobs able to be performed by the same worker on different machines where each job could perfectly fit in the Idle Gap of the others this problem would be optimally solved.

In order to understand which jobs best fit together, an Increment Table was calculated with it's results being the time that the worker has to wait before he can work on the next assigned job.

In the following figure 4.3 this concept is illustrated. Two jobs are presented and one worker is assigned to perform both. Since the Idle Gap of the first operation matches perfectly with the work time of the second, the total idle time of the worker in this operation is zero (i.e. the increment is zero).

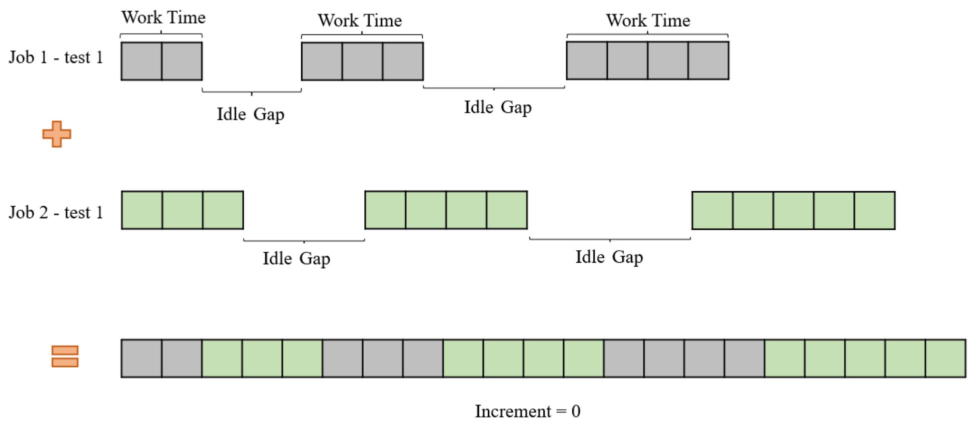


Figure 4.3: Puzzle Piece concept

In this particular example, considering these are the only two available jobs, the increment table can be seen in table 4.3. As demonstrated, test 1 of job 1 has a zero increment to perform the test 1 of job 2. However, the reverse would mean an increment of 19 time instances. This illustrates the importance of allocating compatible jobs together.

Table 4.3: Increment Table

	1.1	2.1
1.1	–	0
2.1	19	–

In order to select and combine the jobs with the least amount of increment time together the reverse of the previous method (IPJS) must be performed. Firstly, the algorithm has to select a worker (respecting the prohibition conditions as well) and then it chooses a machine where this worker can perform at. Afterwards, it obtains the pool of available jobs that both the machine and worker can execute. After this process is completed, depending on the number of times that workers has been chosen before, several different ramifications exist which ultimately leads to the assignment of the chosen job. This is stored in a variable called ”Worker Number”

The ”Work Number” variable shows the number of times a certain worker was already chosen in the assignment cycle. This variable increases to a maximum of two, therefore, when the ”Work Number” surpasses this value it returns to the number one. This means that the increment cycle of the initial solution is capped to assign two workers in a row. These parameters were implemented, as the initial solution is assumed to be improved by the genetic algorithm later on. Increasing the complexity of the algorithm on the Initial Solution would lead to increases in computational time and would risk falling into local optimums. Flowcharts, 4.4 and 4.5 illustrate the Incremental Initial Population with Worker Selection (IIPWS) procedure.

The machine and worker routine is a set of condition cycles intended to qualify or disqualify the reason behind not having any available machines to be chosen. The deeper the layer of questioning, the upper the cycle returns to in the IIPWS. Figure 4.6 illustrates this routine.

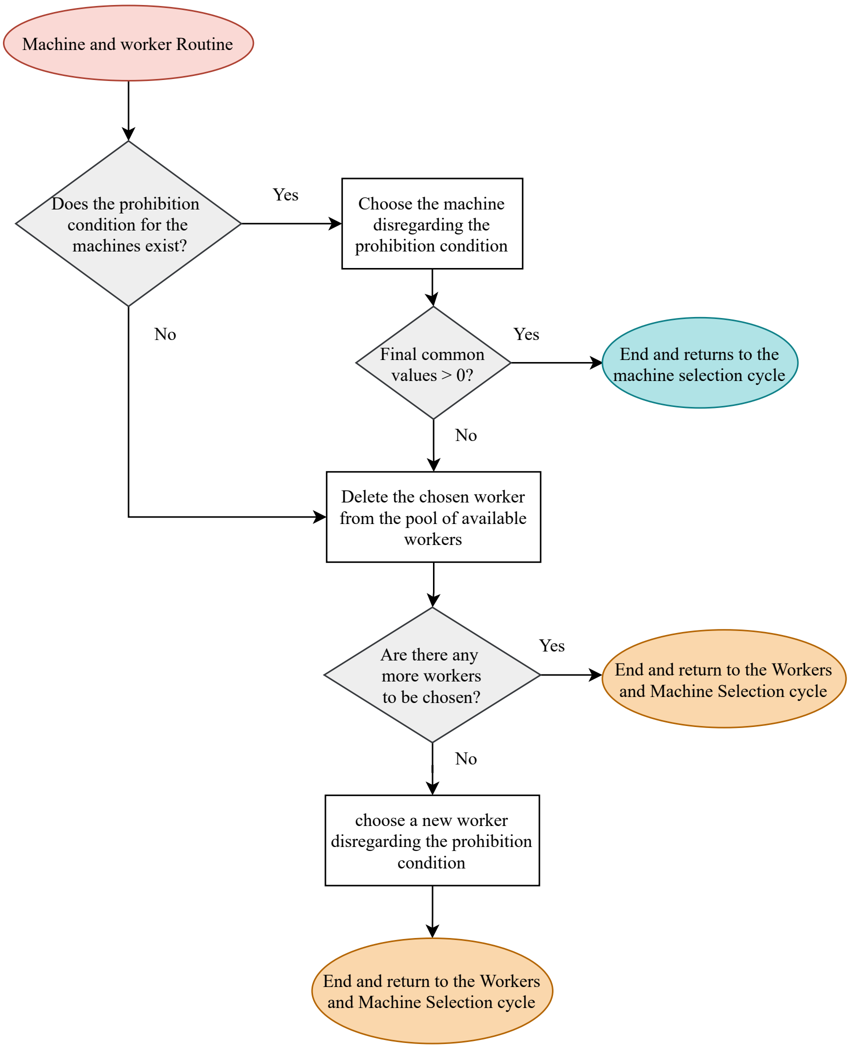


Figure 4.6: Machine and Worker Routine

4.3 Encoding

A three-layered encoding mechanism is proposed based on encoding type 1 and 4 (section 3.1.1). This encoding mechanism was implemented, as it indicates which test is being performed in each job, whilst having the same three-layered chromosome as encoding type 4. The implemented encoding

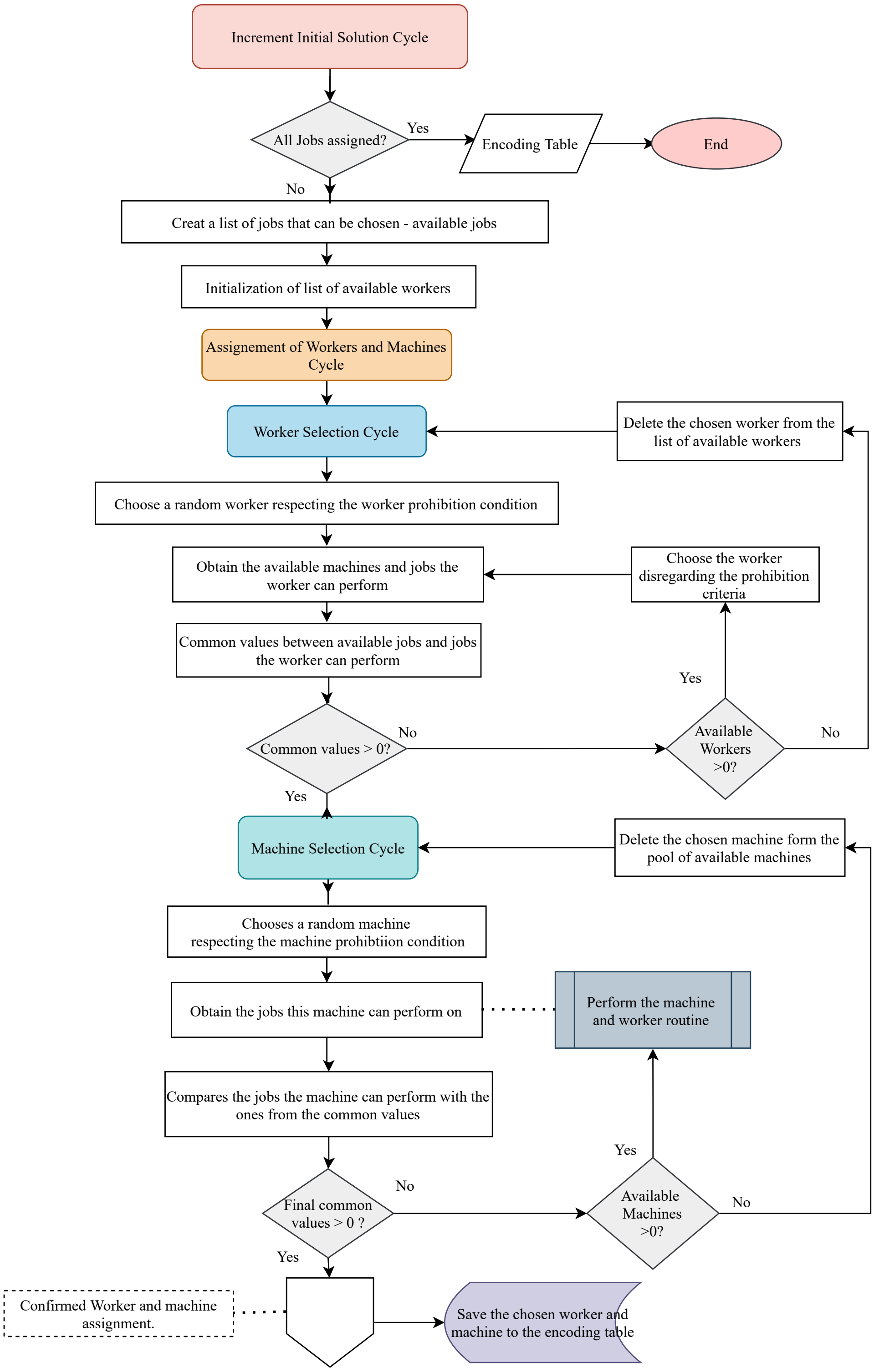


Figure 4.4: Incremental Initial Population with Worker Selection flowchart - part one

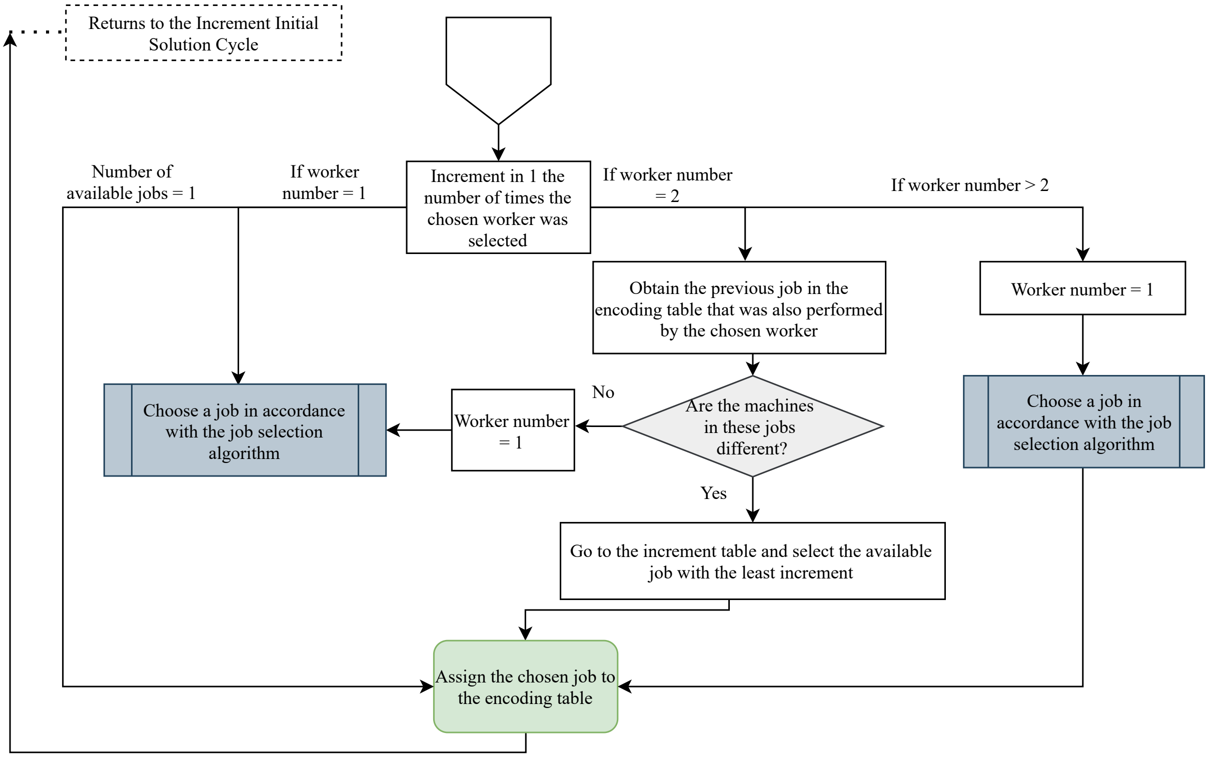


Figure 4.5: Incremental Initial Population with Worker Selection flowchart - part two

scheme has therefore, three chromosomes: the operation sequence chromosome (OS), representing the process sequencing of operations; the machine selection chromosome (MS), which represents the allocation of machines and the Analysts selection chromosome (AS), representing the worker assignment layer. The number of genes in the chromosome equals the total number of operations in all jobs. This representation forces the algorithm to keep the same assignment of resources when a test changes it's position respecting the allocation of machines and workers.

Considering an example with 4 Jobs, 2 Analysts and 7 machines, the following table Table 4.4 represents a possible configuration for the encoding mechanism.

Table 4.4: Three layered Encoding Chromosome

Operation Sequence (OS)	1.1	3.1	2.1	3.2	4.1
Machine Selection (MS)	2	3	5	4	6
Worker Selection (AS)	1	1	1	2	1

In Table 4.4 in the Operation Sequence (OS) chromosome, the unit number represents the job and the decimal number represents the test of that job. More specifically, the forth gene in the OS chromosome is 3.2. This means the test 2 of job 3 is being allocated to this particular position.

The encoding process is performed with the initial population creation and following it is the decodification process.

4.4 Decoding

The Decoding process can be divided into four steps: firstly, define the temporal vectors of the machines and workers (Worker and Machine available time). The vectors were considered to have a resolution of 0.1 representing the workers' and machine times necessary in each operation. This was an approximation done to the problem in order to allow the decodification of the tasks. Secondly, select the operations from the OS vector individually from left to right and obtain the corresponding machine and analyst. Afterwards, define a third temporal vector of the selected job, the operation work times vector with the same 0.1 time resolution. Subsequently, compare the vectors and place the operation as early as possible. Update the temporal vectors and repeat until all jobs are assigned. The decoding process with the temporal vectors is exemplified in figure 4.7 for two distinct consecutive operations where the machine is not repeated.

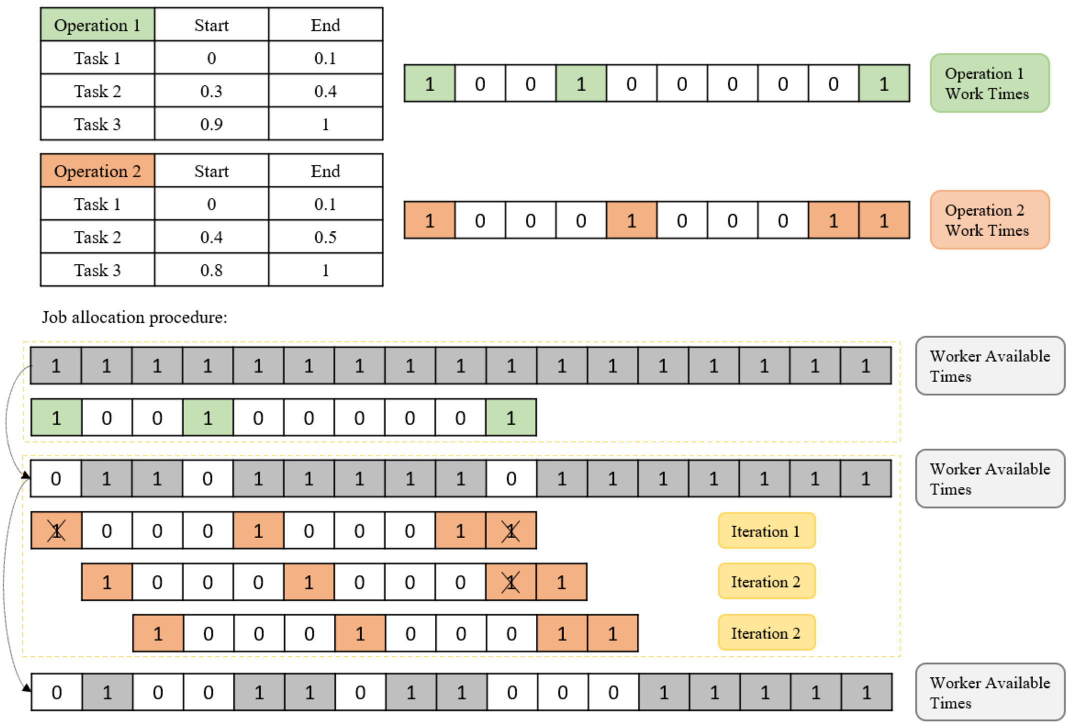


Figure 4.7: Job allocation example with two consecutive operations

4.5 Fitness Evaluation

Following the decoding process, the time at which each job is finished is calculated, so that, the fitness value of the individual can be calculated. In figureTh 4.8 this procedure is illustrated:

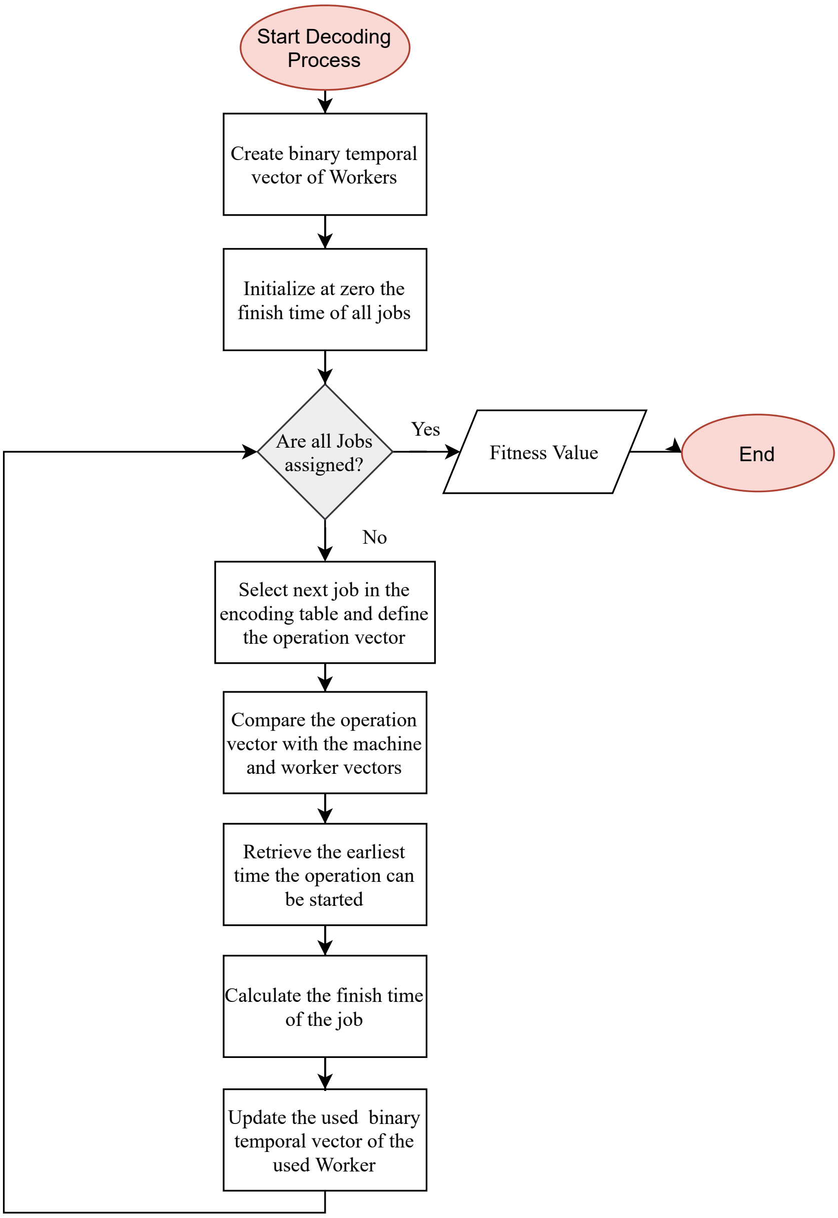


Figure 4.8: Fitness Evaluation - Flowchart

4.6 Selection

In the existing algorithm the Selection phase is achieved through an Elitism operation (which starts once the number of iterations is superior 1/8 of the total population) followed by a Tournament Selection. The Elitism procedure ensures that the best (alpha) solutions are present in the parent population and are not discarded by the Tournament Selection. The delay added to the elitism is to guarantee that the solution has been improved prior to defining any elite population. This aids in eliminating local optimums at the beginning of the algorithm where the solution presents worse fitness values. The Tournament Selection chooses five random solutions from the main population and selects the best one to be added to crossover population. This procedure is repeated until the final population has a certain predefined amount of tournament winning individuals. The bigger the population selected for the tournaments, the less likely it is for weaker individuals to be selected. This is because, if a weak individual is selected to be in a tournament, there is a higher probability that a stronger individual is also in that tournament. Consequentially this implementation mainly chooses the best population to be featured in the Crossover while also giving a chance to the weaker individuals to be present. The lower the number of individuals in a tournament the higher variety of solutions available for crossover, thus reducing the convergence speed of the algorithm.

Figure 4.9 illustrates this two methods used in the selection phase.

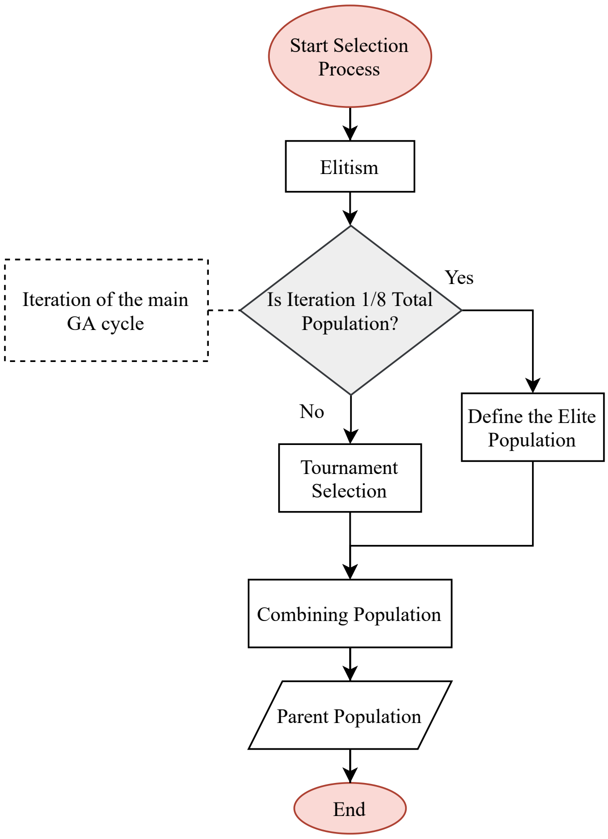


Figure 4.9: Techniques used in the Selection phase - Flowchart

4.7 Crossover

The novel implemented approach explores three different crossover operations: Single Point Crossover, Double Point Crossover and MPX crossover. As previously utilised in the Population Initialization phase (4.2) weights were implemented to account for different probabilities of choosing a certain path. In the next chapter this weight selection is compared and studied. The following figure 4.10 illustrates this phase:

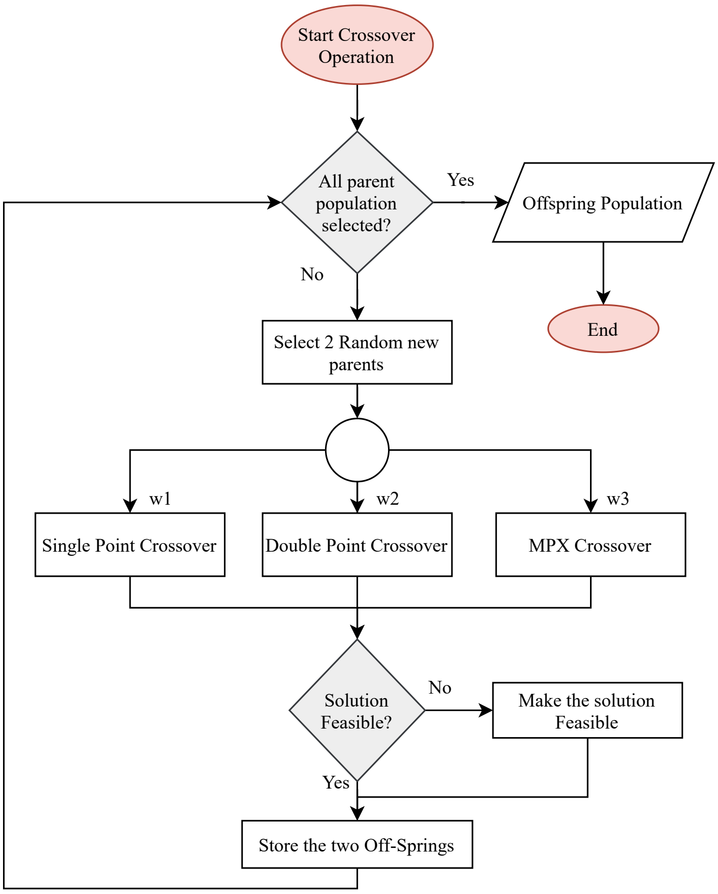


Figure 4.10: Crossover - Flowchart

4.7.1 Single Point Crossover

The Single Point Crossover is performed by firstly, choosing a random point in both parents genes. This divides the parents chromosomes in two sections identical sections. Afterwards, the crossover is completed by swapping parents chromosomes right sides. The following figure 4.11 illustrates this Crossover operation:

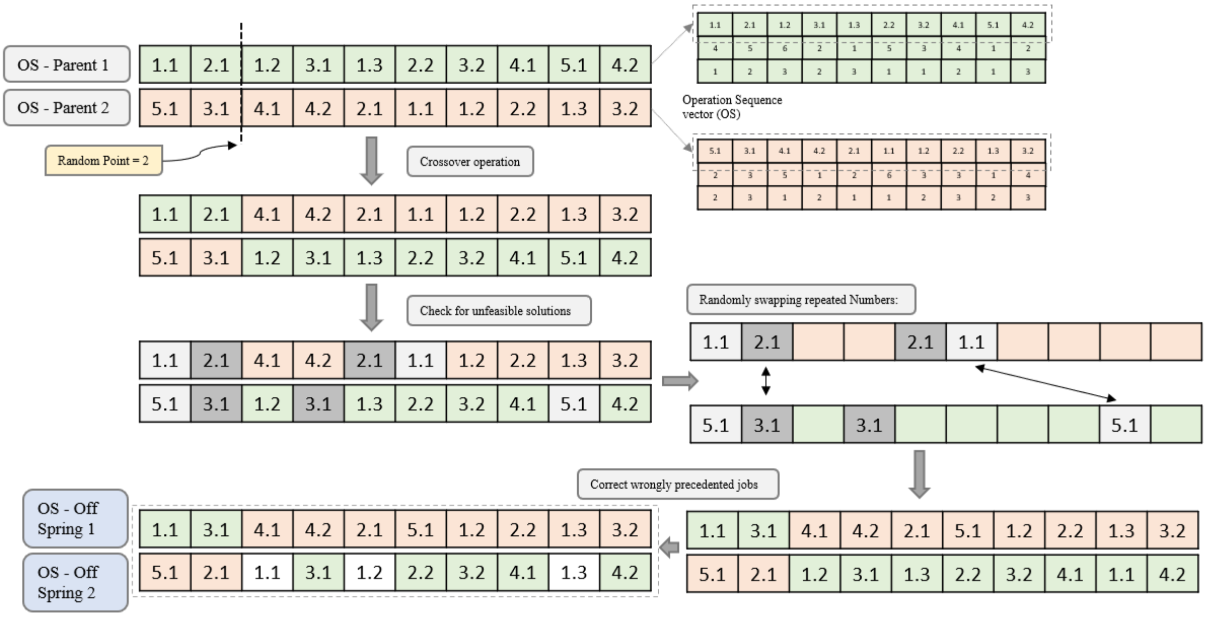


Figure 4.11: Single Point Crossover example

4.7.2 Double Point Crossover

In the Double Point Crossover, two different random numbers (in the chromosome length) are chosen which correspond to a certain interval gap in the parents genes. This middle section is swapped between the two to produce both off-springs. The following figure 4.12 illustrates this Crossover operation:

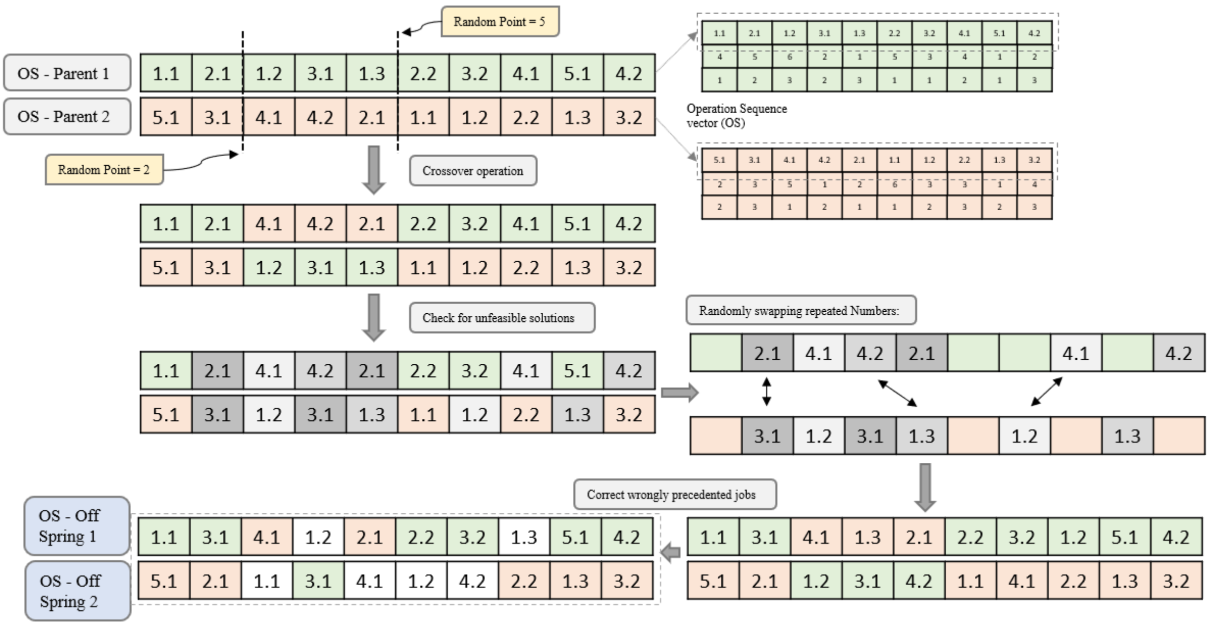


Figure 4.12: Double Point Crossover example

4.7.3 MPX Crossover

The MPX Crossover was based on [36] and is responsible for the change in the allocation of workers and machines. It revolves around the creation of a random binary vector with the same length of the chromosomes, which, depending on the value it assumes on a certain gene, uses the machines and workers from one of the parents to assemble the off-springs. The procedure is the following: assign to the offspring with the same number of the parent the same order of jobs. Going from left to right until the end of the chromosome, retrieve the number of the binary value on that position. If the value of the binary vector is equal to one, parent 1 will assign offspring 2 with the machine and worker he has been assigned to the job the offspring 2 has in that position. The opposite happens for parent 2 and offspring 1. If the value on the random vector is equal to zero, parent 1 assigns the machine and worker he has for that job to the offspring of the same number. Likewise for the second parent and offspring.

The following figure 4.13 illustrates this Crossover operation:

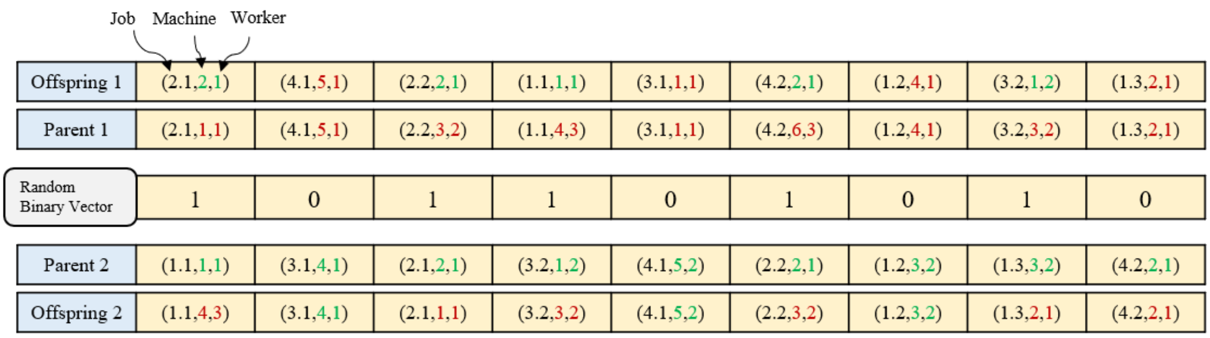


Figure 4.13: MPX Crossover example

4.8 Mutation

Offspring population have a certain chance to mutate. The mutation phase starts with a Shift Mutation (section 4.8.1) based on [4] followed by an "Intelligent Mutation" [21] (section 4.8.2). Depending on a certain predetermined probability, a machine and worker mutation is performed (section 4.8.3). Afterwards, the non-mutated off-springs are combined with the mutated ones to form the final off-springs population, as it can be seen in the following figure 4.14

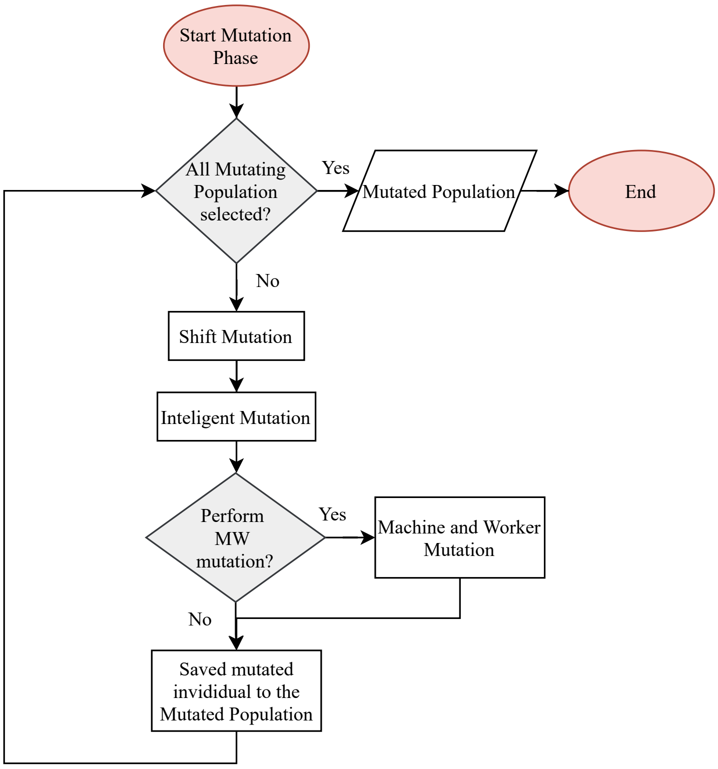


Figure 4.14: Mutation - Flowchart

4.8.1 Shift mutation

The Shift mutation resembles the Double Point Crossover 4.7.2 where two different random numbers with the length of the chromosome are firstly obtained. Each number will correspond to a gene (with a job, worker and machine) and will swap positions with the other number's position. Afterwards, feasibility and precedence checking is required.

4.8.2 Intelligent mutation

The concept behind the Intelligent mutation is to select one of the operations from the machine with maximum workload and assign it to the least used machine. If the machine is not compatible no mutation occurs.

4.8.3 Worker and Machine mutation

This mutation operation uses the same shifted genes from the Shift Mutation 4.8.1. Here both machines and workers are changed to new compatible workforce to perform that job. If no compatibility exists no change is performed.

4.9 Variable Neighbourhood Search

The logic behind Variable Neighbourhood Search (VNS) is to explore neighborhoods of a current incumbent solution. If the new solution proves fitter than the previous one, then the algorithm selects it as the incumbent solution and explores further neighbor solutions until a termination criteria is achieved. This ”exploration” is usually performed by operations similar to the ones implemented in the mutation phase in section 4.8. VNS was implemented in an effort to improve GA’s slow convergence speed due to unguided mutations.

In the implemented VNS, 4 neighborhood structures were implemented: Exchange, Replace, Change and Intelligent structure. Different search algorithms were implemented as it allows for different neighborhood searches which, can potentially improve the convergence speed of the algorithm thus reducing the makespan.

The Exchange structure is analogous to the same exact principle as the shift mutation. The Change structure is equivalent to the Worker and Machine Mutation and the Intelligent structure follows the same logic as the one with the same name in the Mutation phase. The Replace mutation consists of selecting two different random numbers within the chromosome length which consists of two different genes. Subsequently, the gene position that corresponds to the smallest number is placed in the position on the encoding table of the biggest number. The gene in the newly occupied position moves one position backwards as well as the rest of the genes until no more overlapping occurs.

The following implemented algorithm is the following:

Algorithm 4.1: Variable Neighbourhood Search (VNS)

```
Data: Parent and Offspring Population
Result: VNS population
1: Sort the population;
2: Select the best percentage of individuals to perform VNS (VNS population);
3: Obtain individual x;
4: while Not all individuals in the VNS population had undergone the VNS do
    5: Set: a = 0; b = 0; c = 0; d = 0; t = 0;
    6: for t = 1 -t-> iteration do
        if a = 0 or (a = b = c = d = 1) then
            Produce new individual y by Swap operation. Evaluate the individual y;
            if y is better than x then
                x = y; a = 0; t = 0;
            else
                a = 1 - a; t = t + 1;
                if b = c = d = 1 then
                    b = c = d = 0
        if b = 0 then
            Generate new solution y with Insert. Evaluate the individual y;
            if y is better then x then
                let x = y, b = 0, t = 0;
            else
                b = 1 - b; t = t + 1;
        if c = 0 then
            Produce new individual y with Assign. Evaluate the individual y;
            if y is better then x then
                let x = y, c = 0, t = 0;
            else
                c = 1 - c; t = t + 1;
        if d = 0 then
            Generate new solution y with Intelligent. Evaluate the individual y;
            if y is better then x then
                let x = y, d = 0, t = 0;
            else
                d = 1 - d; t = t + 1;
        if t > iteration/4 then
            Terminate
    7: Return individual x;
```

4.10 Dynamic Termination Criteria

The termination criteria was implemented in the event the algorithm did not yield any improvements after N iterations. Since this lack of improvement may be related to a local optima, a procedure was implemented to avoid the algorithm being trapped in one. In this case, a counter of the successive non improving iterations was implemented, that when it surpasses $\frac{1}{4}$ of the termination criteria variable, the percentage of offspring and the mutation percentages increases to 80% and 50% respectively. Also if the VNS is active it increases to 30% of the best population. Afterwards, an intelligent mutation is performed to all the population. If the total completion time of the instance is reduced, this new individual replaces the previous one. If the algorithm improves these parameters return to the normal values and the counter is returns to zero. The algorithm may end after a predefined number of non-improving consecutive iterations is reached (termination criteria). The algorithm is also limited to 3h of total computational time. The following algorithm (2) illustrates this logic:

Algorithm 4.2: Termination Criteria

Pr

```
Data: Best Solution
Result: Continuation or Termination of the algorithm
while Computational time l<3h do
    if algorithm did not improve from last iteration then
        number of non-improving iterations += 1 (counter);
        if counter ⩾ 1/4 termination criteria then
            Mutation coefficients = 50%;
            Percentage of best population in VNS = 30%;
            Percentage of Offspring = 80%;
        if counter ⩾ termination criteria then
            terminate algorithm;
    else
        counter = 0;
        All values return to the original state;
```

5

Results

Contents

5.1 Description of the Experiments

The experiments developed encompass different configurations of the QC labs scheduling problem. To allow for realistic data, a study by Martins et al. [2] was conducted on real quality control laboratory and an instance generator was developed. The instances used in this work are the same ones used by [2]. The instance generator aims to mimic the conditions faced on a regular working day in this Quality Control Laboratory environment. The instance generation approach followed makes use of the sample type (job type) concept, which has direct relation to what might happen in real life scenario in QC labs. Each combination of product and source (e.g. raw material, final product) usually results on a predetermined set of tests (operations) that need to be completed. This way, jobs come from a set of predetermined possible job types, each characterised by a number of operations and their processing times.

Regarding the aforementioned job types, J, three are considered in all experiments. For each job type, the number of operations is determined using a discrete uniform distribution (DUD), ranging from 1 to 3. Each operation has a processing time p_{ij} ranging from 1 to 5 and the worker tasks start from three time points: the start of the operation, at 30% of operation processing and at 90% of operation processing. The duration of the worker tasks are: 5%, 10% and 10% of the total processing time, representing the setup, intermediate and disassembly/data processing tasks.

The parameters that characterise experiments are the number of jobs n, number of machines m, number of workers w and the flexibility. Experiments have been developed for 5, 10 and 70 jobs. The smaller instances have been used to facilitate the implementation of the algorithm and are not representative of the QC lab dimension. The medium 12 instances of 10 jobs represent the daily workload of a QC laboratory. The larger 70 job instances mimic the realistic QC labs weekly scheduling problem. The number of machines is set at 7, representing different types of equipment present at laboratories. For this thesis, the implementation developed is flexible. i.e., it can be used for any number of machines.

The number of workers can be 2, 3 and 7, representing cases that are respectively, worker restricted, balanced and machine restricted. The flexibility parameter is used to compute the machines and workers that are able to carry out a certain operation. When generating the instance, each one of the workers and machines has a probability equal to their flexibility of being eligible to carry out the operation. Experiments are done for flexibilities of 30% and 60%, encompassing cases where machines and workers have less or more general competences. If more than one machine and worker can perform a certain job, the time they take is the same, i.e., there is no heterogeneity between workers and machines that can perform the same operation.

The experiments developed are summarised in Table 5.1. Three replications of each experiment have been generated to improve the reliability of results. With n taking three possible values, w taking three possible values and the flexibility with two possible values, the total number of instances with different experimental parameters is eighteen. Considering that each of these instances have three distinct unique parameter variations, this is three replications, there are a total of fifty-four distinct experiments. [2]

All experiments were performed on a computer running Windows 10 with an Intel Core i5-8265U processor (1.60GHz base frequency) and 8GB of RAM.

Table 5.1: Parameters for the experiments designed [2]

Parameters		Values
Job generation parameters		
Job Types	N	3
Operations in job	q_j	$DUD(1,3)$
Operation processing time	p_{ij}	$DUD(1,5)$
Starting point of worker tasks	1	$[0, 0.3p_{ij}, 0.9p_{ij}]$
Duration of worker tasks	p^d_{ijs}	$[0.05p_{ij}, 0.1p_{ij}, 0.1p_{ij}]$
Experiment Parameters		
Number of jobs	n	$[5, 7, 10]$
Number of machines	m	7
Number of workers	w	$[2, 3, 7]$
Flexibility	n	$[0.3, 0, 6]$

5.2 General GA Parameters

The general genetic algorithm parameters are considered to be the essential parameters, present in every GA. The correct tuning of this constants is not a trivial task, since these variables are often co-dependent. This section will cover how the tuning of these parameters could potentially improve the algorithm's results and present the values considered in this dissertation. These tests ran without VNS and a sample instance was considered (70 jobs, 7 machines, 3 workers, 0.3 flexibility and number 0) since to understand the degree of convergence of the algorithm the tests need to run for long periods of time, therefore, computing all 18 instances of 70 jobs proved to be unfeasible. The results are the average of the the best result in 5 runs.

5.2.1 Population Size

Regarding the population size tuning, a total of 50 iterations were considered and kept constant as well as an offspring percentage of 70% and a mutation chance of 50%. The average standard deviation of the results was 57.2 hours. The results are presented in figure 5.1 below:

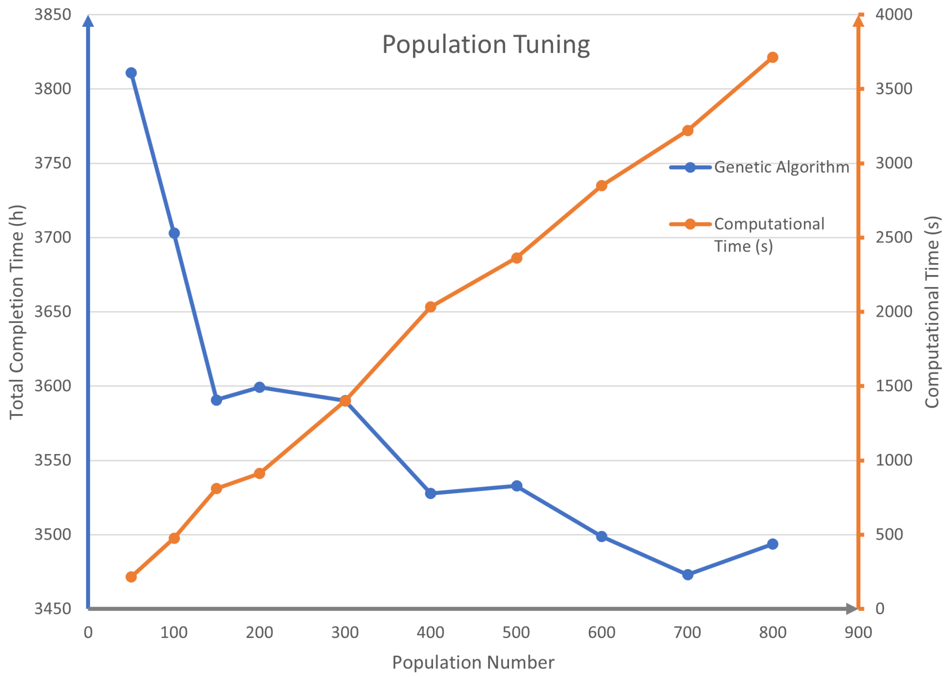


Figure 5.1: Study of the Total Completion Time with different population sizes

From 5.1 the most suitable population numbers are 150 with an average of 811 seconds of computational time, 400 with 2032 seconds and 700 with 3219 seconds.

Since the final results are meant to run the instances for 3h (10800 seconds), with a population of 700 individuals, the total number of iterations could only be about 3.35 times the number of iterations used in this tests (meaning roughly 160 iterations) which is a low number to allow a proper convergence of the algorithm. As for the 400 individuals it allows for roughly 5 times the 50 iterations, i.e., 250 total iterations. For the 150 individuals it allows up to 13 times the 50 iterations (i.e., around 650 iterations). Since the objective is to obtain the best possible result in 3h the 400 off population was chosen since it provides a good compromise between the results and the computational time.

5.2.2 Offspring Percentage

Regarding the offspring percentage tuning, apart from the other parameters, a total of 100 iterations with a total population of 400 was considered. The percentage of population to be selected for the offspring operation was changed from 10 to 100%. The results can be seen in the following figure 5.2, in which the average standard deviation is 62.4 hours.

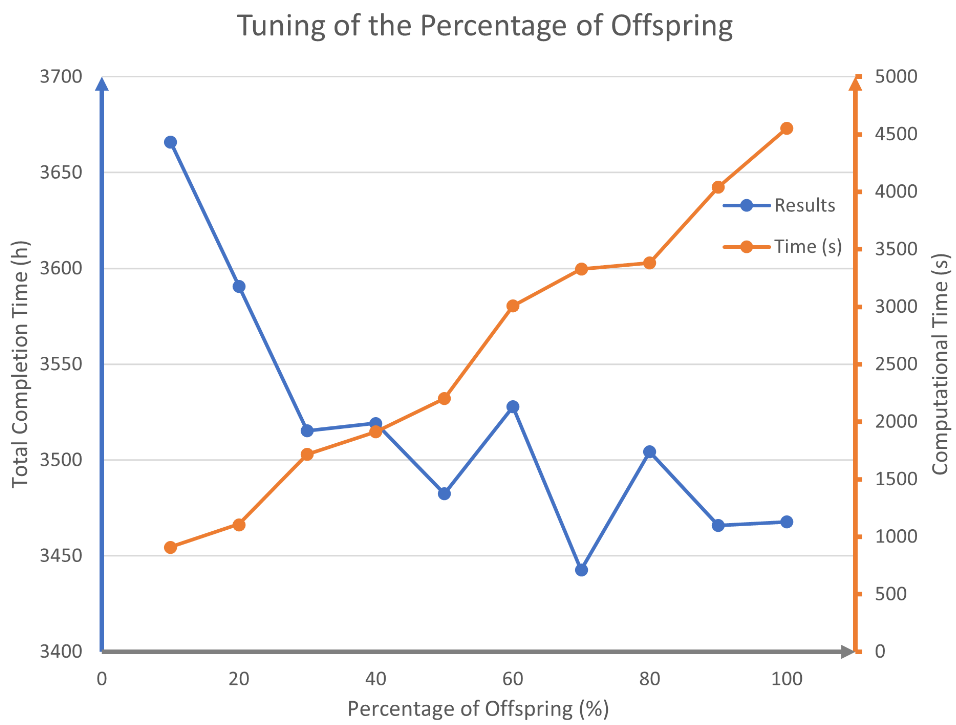


Figure 5.2: Study of the Total Completion Time with different percentage of Offspring

From figure 5.2 one can remark that the values presenting the best results for the offspring percentage are the 50% and 70%. Since the objective is to obtain the best possible results in 3h, the latter value was considered.

5.2.3 Mutation Percentage

For the percentage of mutation tuning, a total of 50 iterations were considered while the other parameters were the ones obtained from the previous tuning (400 of population and 70% of offspring). The following figure 5.3 shows the results in which the average standard deviation is 67.4 hours.

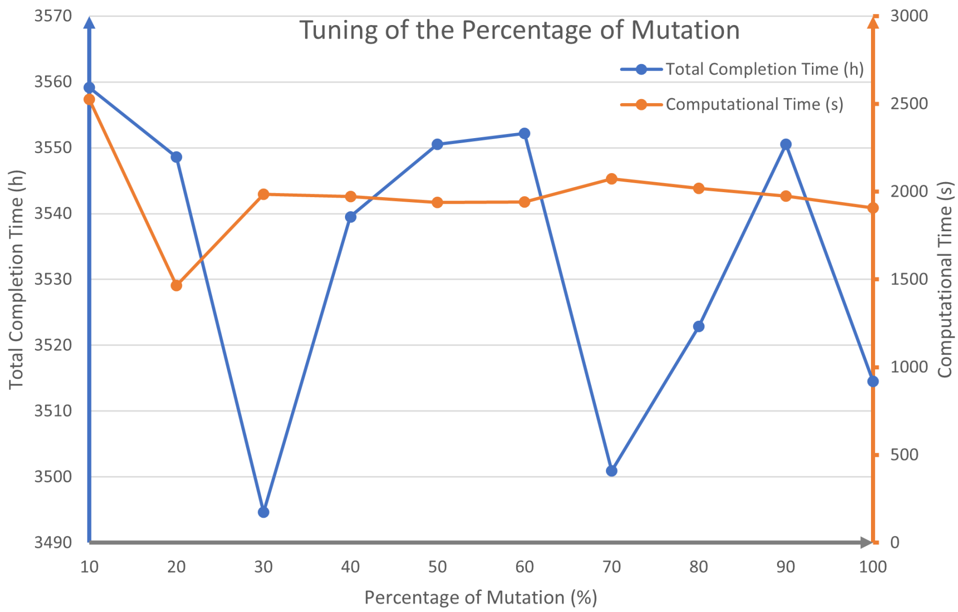


Figure 5.3: Study of the Total Completion Time with different percentage of mutation

From figure 5.3 one can conclude that from the study the mutation is not very relevant for this instance (results differ by 1%). Nevertheless the best percentage of mutation is 30% as it induces the least total completion time in the model instance.

5.3 Initial Population

In order to obtain the best possible results, every weight must be tuned. Therefore the adjustments start in the initial population. The goal is to obtain the best solutions without falling into a local optima.

5.3.1 Job Selection Procedure

Tests were performed to the Job Selection Procedure (section 4.2.1) in order to verify which combination of weights (w_1, w_2, w_3) proved better results. The following tables 5.2 and 5.3 contain the average values followed by the respective standard deviation returned by the algorithm for the 70 job instances. These were obtained from a population of 500 individuals. In green is the best value and in yellow the second best for the same instance.

Table 5.2: Weight Tuning part 1

Nº Jobs	Wor-kers	Flex	Rep	100|0|0	0|100|0	0|0|100	40|30|30	60|20|20	60|30|10
70	2	0.3	0	5277 +/-211	5558 +/-225	6428 +/-306	5957 +/-261	5786 +/-254	5696 +/-255
			1	7369 +/-283	7492 +/-317	8250 +/-392	8095 +/-319	7860 +/-324	7750 +/-277
			2	4541 +/-137	4568 +/-152	4938 +/-176	4629 +/-145	4624 +/-150	4583 +/-142
		0.6	0	5194 +/-206	5303 +/-195	6243 +/-239	5336 +/-202	5298 +/-197	5212 +/-188
			1	5651 +/-256	5923 +/-310	6627 +/-248	5773 +/-260	5696 +/-247	5649 +/-255
			2	5681 +/-276	6240 +/-305	6690 +/-290	6278 +/-307	6104 +/-305	6000 +/-287
	3	0.3	0	4615 +/-237	4208 +/-235	5765 +/-252	5314 +/-310	4983 +/-272	4930 +/-285
			1	13050 +/-825	13645 +/-873	14596 +/-843	13920 +/-924	13695 +/-879	13610 +/-906
			2	6208 +/ 256	7211 +/-341	7195 +/-239	6909 +/-297	6710 +/-288	6750 +/-313
		0.6	0	3695 +/-176	4104 +/-224	4151 +/-165	4024 +/-182	3961 +/-180	3961 +/-182
			1	3863 +/-187	4300 +/-247	4325 +/-188	4108 +/-230	4033 +/-192	4004 +/-194
			2	2718 +/-163	2937 +/-205	3064 +/-203	2955 +/-199	2892 +/-193	2862 +/-192
	7	0.3	0	4362 +/-273	4877 +/-346	4880 +/-304	4708 +/-335	4606 +/-296	4596 +/-304
			1	6612 +/-450	6835 +/-482	6957 +/-456	6909 +/-509	6834 +/-456	6794 +/-458
			2	3627 +/-177	3739 +/-212	3805 +/-171	3655 +/-204	3608 +/-180	3600 +/-191
		0.6	0	5680 +/-281	5427 +/-269	6376 +/-303	6082 +/-304	5937 +/-312	5878 +/-286
			1	2831 +/-174	3324 +/-234	3480 +/-213	3388 +/-221	3220 +/-219	3156 +/-217
			2	4564 +/-286	5180 +/-279	5281 +/-281	5106 +/-317	4960 +/-290	4944 +/-301
Total				**95536**	**100871**	**109048**	**103146**	**100805**	**99976**

Table 5.3: Weight Tuning part 2

Nº Jobs	Wor-kers	Flex	Rep	60|10|30	80|10|10	90|5|5	95|5|0	95|0|5	98|2|0
70	2	0.3	0	5815 +/-256	5549 +/-248	5413 +/-233	5341 +/-244	5334 +/-215	5289 +/-228
			1	7938 +/-344	7603 +/-312	7504 +/-291	7441 +/-288	7466 +/-314	7384 +/-287
			2	4683 +/-156	4582 +/-155	4565 +/-150	4553 +/-143	4547 +/-144	4548 +/-145
		0.6	0	5407 +/-211	5254 +/-197	5219 +/-191	5180 +/-189	5217 +/-212	5194 +/-206
			1	5821 +/-264	5632 +/-255	5626 +/-245	5599 +/-258	5716 +/-248	5641 +/-256
			2	6204 +/-295	5925 +/-278	5797 +/-275	5703 +/-264	5764 +/-285	5676 +/- 275
	3	0.3	0	5038 +/-262	4762 +/-255	4656 +/-232	4604 +/-236	4634 +/-239	4602 +/-218
			1	13895 +/-860	13462 +/-897	13252 +/-871	13129 +/-827	13206 +/-850	13101 +/-838
			2	6751 +/-260	6516 +/-296	6355 +/-284	6275 +/- 251	6289 +/-261	6235 +/-272
		0.6	0	3987 +/-177	3871 +/-172	3792 +/-168	3755 +/-181	3752 +/-195	3706 +/-177
			1	4100 +/-191	3986 +/-190	3931 +/-189	3897 +/-193	3931 +/-179	3886 +/-206
			2	2896 +/-186	2818 +/-180	2786 +/-173	2752 +/-180	2737 +/-167	2731 +/-167
	7	0.3	0	4613 +/-309	4500 +/-282	4439 +/-272	4411 +/-289	4417 +/-274	4411 +/-268
			1	6805 +/-468	6727 +/-460	6689 +/-465	6619 +/-485	6651 +/-455	6633 +/-1437
			2	3657 +/-171	3620 +/-178	3626 +/-172	3586 +/-155	3631 +/-170	3609 +/-166
		0.6	0	5979 +/-300	5826 +/-306	5749 +/-287	5667 +/-276	5707 +/-300	5683 +/-285
			1	3238 +/-217	3047 +/-205	2951 +/-202	2869 +/-196	2910 +/-190	2853 +/-185
			2	4974 +/-299	4812 +/-288	4682 +/-299	4627 +/-196	4590 +/-256	4579 +/-271
Total				**101799**	**98488**	**97031**	**96009**	**70093**	**95759**

The reasoning behind the different experimented weights was: firstly, assume the maximum probability to each path and rank them from best to worst. This likely means that the more the best paths are followed by the algorithm the better the results are. Therefore the following tests were conducted bearing in mind that w_1 and w_2 proved the best and second best results (respectively).

It can be noticed that when the weight one, w_1, has 100% of probability of being chosen, which corresponds to the random initialization, the results proved to be the best for nearly all the instances.

It is worth mentioning that the 70 Job instance is used throughout all tuning tests in this work as it has the most job allocation and consequentially, more reliability, and is the target instance of study for this work.

5.3.2 Prohibition Condition Study

The prohibition Condition as explained in section 4.2 assures that there is a limit for allocating a certain X number of workers and Y number of machines in a row. In this section, a tuning of these parameters is performed the same way as it was done in the section 5.3.1 above.

Isolated Example with and without Prohibition Conditions

Firstly, to prove that the prohibition condition improves the results, let's consider a small instance with 5 jobs, 3 workers and flexibility 0.6. A possible first configuration for the encoding table without any Prohibition Conditions applied could be:

Table 5.4: Example with no Prohibition Condition applied

1.1	5.1	1.2	3.1	5.2	4.1	2.1	2.2	3.2	3.3	4.2
5	1	7	7	7	5	5	1	7	2	7
1	3	3	2	3	1	3	1	1	2	3

The present configuration 5.4 has a total completion time of 58 time units. Now, if the reasoning provided is correct, if one applies the prohibition condition to encoding table, with both workers and machine not able to be chosen more than 2 times in a row, the fitness value should improve or in the very least stay the same. In the present example, on the job 5.2, machine 7 needs to be replaced by another suitable machine. The suitable machines for job 5.2. are: Machines 1, 2, 4 or 5. Applying the decoding and using the same objective function, the total completion time for machine 1,2 and 4 are, respectively: 44.4, 44.4 and 44.4 time units. For machine 5 it must be taken into account that this creates another prohibition condition in job 2.1 as it can be seen in the table below 5.5:

Table 5.5: Prohibition applied in job 2.1

1.1	5.1	1.2	3.1	5.2	4.1	2.1	2.2	3.2	3.3	4.2
5	1	7	7	5	5	5	1	7	2	7
1	3	3	2	3	1	3	1	1	2	3

The present configuration in table 5.5 has a total completion time of 49.4 time units. Following the same reasoning as before, the available machines for job 2.1 are: 1, 3, 4 or 7. Since changing to any of these values does not induce any other prohibition conditions, the total completion time for these machines are, respectively: 41.4, 40.5, 40.5 and 40.5, proving that introducing a prohibition condition on the machines improves the initial solution.

Now let's consider another possible first configuration for the encoding table without any Prohibition Conditions where a worker is chosen 3 times in a row. A possible configuration for the encoding table could be:

Table 5.6: Example with no Prohibition Condition applied - worker

1.1	5.1	1.2	3.1	5.2	4.1	2.1	2.2	3.2	3.3	4.2
5	1	7	7	5	5	5	1	7	2	7
1	1	1	2	3	1	3	1	1	2	3

The present configuration in table 5.6 has a total completion time of 49.4 time units. With the same reasoning, another possible worker to perform job 1.2 is worker 3 which makes the total completion time equal to 40.7 time units, thus improving the solution.

Weight tuning of the Prohibition Conditions

The reasoning for the following experimented weights was: first check if there is some correlation between the number of times the prohibition conditions are fired and the mean results obtained. Secondly, while maintaining one of the conditions fixed (either the worker or the machines) check if by increasing or decreasing the other, it changes the results for better or worse. The results on table 5.7 were obtained by performing the average of five runs on a population size of 500 for each instance. The average standard deviation across all instances was +/- 259.

Table 5.7: Prohibition Condition values tuning

Nº Jobs	Workers	Flex	Rep	GA no Prohibition	GA W = 3 M = 2	GA W = 3 M = 3	GA W = 3 M = 4	GA W = 3 M = 6	GA W = 4 M = 3	GA W = 4 M = 4	GA W = 4 M = 5	GA W = 5 M = 5
70	2	0.3	0	5413.141	4985	5001	5007	5005	5067	5056	5057	5112
			1	7500.56	7316	7356	7384	7378	7378	7381	7384	7392
			2	4558.991	4554	4544	4541	4543	4537	4546	4548	4547
		0.6	0	5211.68	5138	5134	5134	5149	5141	5149	5143	5163
			1	5639.7	5441	5455	5449	5450	5533	5529	5529	5552
			2	5803.67	5642	5648	5650	5648	5637	5665	5643	5661
	3	0.3	0	4675.081	4521	4548	4593	4585	4546	4568	4570	4594
			1	13412.95	12096	12497	12783	12980	12552	12720	12931	12905
			2	6349.8	6134	6172	6198	6204	6172	6216	6206	6201
		0.6	0	3792.893	3647	3679	3673	3674	3676	3683	3680	3680
			1	3923.9	3850	3854	3876	3870	3871	3863	3862	3875
			2	2763.56	2686	2701	2704	2729	2703	2717	2733	2710
	7	0.3	0	4403.69	4310	4343	4369	4374	4358	4387	4359	4364
			1	6714.08	6502	6606	6609	6598	6597	6579	6610	6594
			2	3623.75	3565	3585	3615	3610	3596	3599	3627	3625
		0.6	0	5738.35	5558	5628	5644	5667	5629	5663	5660	5653
			1	2957.89	2816	2836	2839	2831	2837	2847	2855	2846
			2	4682.125	4512	4538	4529	4553	4532	4520	4552	4543
Average				**5398**	**5182**	**5229**	**5255**	**5269**	**5242**	**5260**	**5275**	**5279**

From table 5.7 it can be noticed that the prohibition condition achieves the best results for W=3 and M=2, improving results by 4% when comparing with a GA with no prohibition condition.

It is worth mentioning that values below $W = 3$ and $M = 2$ were not tested since this could potentially risk falling into local optimum in the very beginning of the algorithm. The values of the prohibitions means it is the value when the prohibition starts. Therefore if $W = 3$, it means that the algorithm will not allow 4 equal workers to be sequentially assigned .

As it can be seen in Appendix B, in tables B.1 and B.2 there is no correlation between the amount of times a certain prohibition is fired and the results on that particular instance.

5.3.3 IP with Job Selection vs Incremental IP with worker selection

The final addition to the Initial Population is the Incremental Initial Population with worker selection as it is explained in section 4.2.2. The following table 5.8 contains a comparison between the average values of the Normal initialization (IPJS) with the average values of this new approach. These are obtained from a population of 500 individuals and the following values are the respective standard deviation. As it happened previously, the best values for each instance are marked in green.

Table 5.8: IP with Job Selection (normal initialization) vs Incremental IP with worker selection (increment initialization)

Nº Jobs	Workers	Flex	Rep	Normal Initialization	Increment Initialization
70	2	0.3	0	5006.7 +/-184	4958.5 +/-192
			1	7324.3 +/-287	7245.2 +/-248
			2	4540.2 +/-143	4579.7 +/-171
		0.6	0	5141.4 +/-182	5184.9 +/-214
			1	5435.3 +/-216	5193.9 +/-183
			2	5600.7 +/-245	5531.8 +/-196
	3	0.3	0	4519.9 +/-203	4900.3 +/-211
			1	12024.1 +/-675	10027.1 +/-566
			2	6108.2 +/-243	6057.6 +/-321
		0.6	0	3655.2 +/-182	3098.7 +/-120
			1	3850.4 +/-177	3677.4 +/-158
			2	2689.9 +/-164	2524.3 +/-141
	7	0.3	0	4331.1 +/-251	4088.6 +/-247
			1	6503 +/-415	5934.4 +/-372
			2	3562.1 +/-153	3572.3 +/-184
		0.6	0	5603.2 +/-275	5141.9 +/-233
			1	2822.8 +/-177	2414.6 +/-112
			2	4490.2 +/-250	4142.5 +/-172
Total:				**93208.7**	**88273.7**

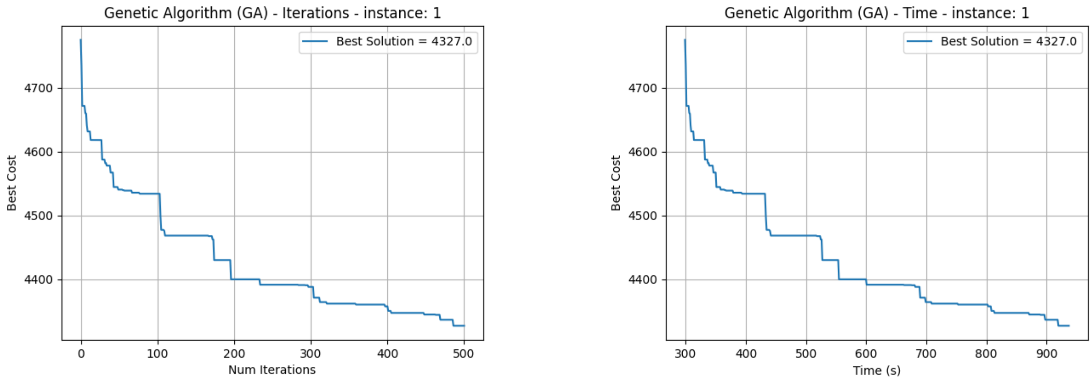
From table 5.8 it can be noticed that for the majority of the instances the Increment Initialization (IPJS) obtains fitter results thus proving the accuracy of this implementation. When comparing the total completion time of all instances, the Increment Initialization improves the results by a margin of 5.3% when comparing to the Normal Initialization.

Convergence Speed

The results from table 5.8 prove that the Increment Initialization reaches fitter solutions in the Initial Population. This section studies whether this solutions are capable of having a faster convergence speed in the final solution of the Genetic Algorithm. In this case, a study was conducted for the first instance where the best initial and final objective functions for both the normal and incremental initialization were registered. The algorithm ran with a population of 10 individuals for 500 iterations with no VNS. The results can be seen in the following table (5.9) and in the figures 5.4 and 5.5:

Table 5.9: Convergence Study

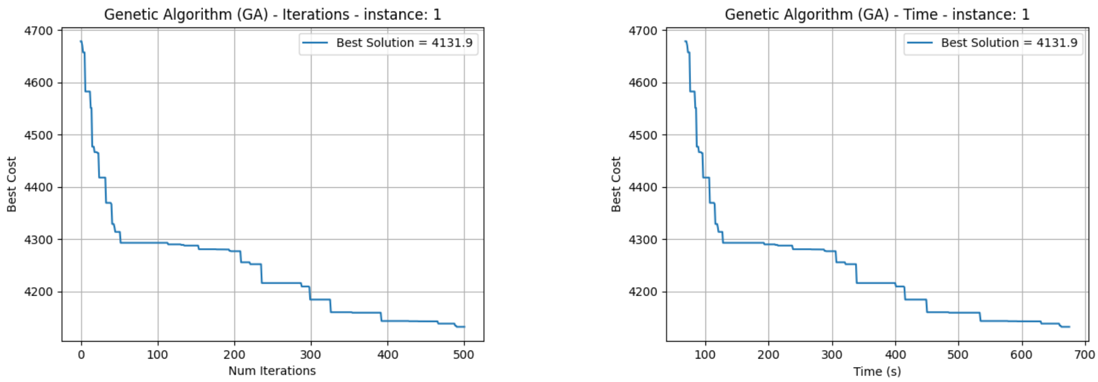
	Best Initial Population Solution	Best Final Solution	Difference (%)
Normal Initialization	4774.9	4327.0	9.38
Increment Initialization	4678.7	4131.9	11.69



(a) Iteration Plot

(b) Time Plot

Figure 5.4: Normal Initialization convergence



(a) Iteration Plot

(b) Time Plot

Figure 5.5: Increment Initialization convergence

As it can be seen, the increment Initialization has a faster convergence speed while also achieving fitter solutions thus proving the strength of this implementation.

5.4 Crossover Tuning

Tests were performed to the Crossover methods (section 4.7) in order to identify which combination of weights (c_1 - Single Point Crossover, c_2 - , Double Point Crossover and c_3 - MPX crossover) proved fitter results. The following tables 5.10 and 5.11 present the best objective function obtained from a population size of 75 with 35 total iterations. In green is the best value and in yellow the second best for the same instance.

Table 5.10: Crossover Tuning part 1

Nº Jobs	Wor-kers	Flex	Rep	100|0 |0	0|100 |0	0|0 |100	30|30 |40	20|20 |60	10|10 |80	5|5 |90	0|50 |50	0|25 |75
70	2	0.3	0	4168	4277	3844	3991	4301	3986	3793	3976	3904
			1	6141	5940	5963	5978	5875	6184	5976	6014	6062
			2	3738	3724	3802	3649	3668	3657	3745	3599	3792
		0.6	0	4448	4162	4303	4159	4303	4263	4161	4192	4226
			1	4756	4660	4298	4498	4250	4353	4193	4302	4195
			2	4558	4502	4485	4587	4496	4456	4406	4449	4474
	3	0.3	0	3638	3746	3675	3441	3698	3634	3607	3712	3645
			1	8343	7873	7180	7881	7801	7802	7446	7670	7298
			2	4942	5042	4827	4868	4896	4790	4788	4880	4810
		0.6	0	2976	2914	2772	2817	2841	2855	2950	2772	2916
			1	3133	3097	3041	3181	3147	3061	3025	3099	3132
			2	2041	2060	1954	2024	1925	1960	1940	1969	1983
	7	0.3	0	3337	3331	3279	3291	3307	3293	3337	3250	3339
			1	4757	4637	4496	4669	4618	4426	4496	4598	4642
			2	2902	2811	2888	2790	2920	2847	2726	2865	2821
		0.6	0	4669	4679	4268	4268	4118	4157	4192	4265	4447
			1	2041	2189	2024	2153	2025	2127	2078	2079	2088
			2	3346	3548	3342	3604	3309	3375	3212	3456	3271
Total				**73931**	**73189**	**70439**	**71849**	**71497**	**71222**	**70071**	**71146**	**71045**

Table 5.11: Crossover Tuning part 2

Nº Jobs	Wor-kers	Flex	Rep	0|15 |85	0|10 |90	10|0 |90	10|5 |85	15|5 |80	20|5 |75	50|0 |50	70|0 |30	70|5 |25	33.3|33.3 |33.4
70	2	0.3	0	4029	3962	3843	3911	3825	3966	3980	4047	4009	3841
			1	5975	6141	6037	6095	6077	6037	6133	6182	6138	6026
			2	3601	3711	3724	3641	3600	3656	3730	3748	3640	3732
		0.6	0	4107	4279	4023	4041	4260	4047	4280	4285	4364	4257
			1	4254	4312	4383	4372	4194	4492	4648	4252	4659	4452
			2	4346	4369	4435	4370	4368	4261	4392	4485	4611	4512
	3	0.3	0	3692	3651	3702	3616	3584	3595	3606	3609	3619	3572
			1	7514	7660	7291	7024	7717	7391	7463	8099	7670	8171
			2	4651	4763	4660	4616	4776	4521	4788	4700	4631	4746
		0.6	0	2872	2978	2823	2738	2903	2856	2813	2953	3005	3052
			1	3015	2945	3033	3032	3049	3061	3090	3222	3055	3018
			2	1949	1908	1937	1957	1957	1929	1990	2106	2002	2051
	7	0.3	0	3285	3241	3276	3322	3302	3306	3269	3260	3337	3274
			1	4649	4644	4498	4582	4556	4574	4568	4656	4662	4569
			2	2806	2804	2804	2841	2810	2818	2763	2908	2828	2861
		0.6	0	4138	4144	4131	4133	4217	4247	4228	4369	4401	4209
			1	2178	2099	2054	2052	2091	2116	2110	2118	2080	2056
			2	3356	3325	3438	3304	3276	3384	3357	3270	3504	3312
Total				**70415**	**70935**	**70093**	**69645**	**70562**	**70256**	**71208**	**72269**	**72214**	**71711**

The reasoning behind the different experimented weights was similar to (5.3.1): firstly, assume the maximum weight to each path and rank them from best to worst. This likely means that the more the best path is followed by the algorithm the better the results are. Therefore, the following tests were conducted bearing in mind that c_3 proved the best results. The next four tests with an incremental increase towards the 3rd weight demonstrate this reasoning and obtain the second best result for the weights: $c_1 = 5$, $c_2 = 5$, $c_3 = 90$. Afterwards, a disruptive weight selection ($c_1 = 0$, $c_2 = 50$, $c_3 = 50$) was done to check whether results would improve. This was followed by a progressive increase in the 3rd weight while marginally changing the other two. The best results were obtained for $c_1 = 10$, $c_2 = 5$, $c_3 = 85$. Afterwards, the remaining tests were attempts to give extra weight to the first path proving the logic that the worthiest weight to increase is the 3rd, corresponding to the MPX crossover.

5.5 Variable Neighborhood Search

5.5.1 Tuning

The parameters to tune in the VNS are the following:

- Percentage of best population to undergo VNS.
- Number of iterations in the VNS.

These must be tuned so that the increase in computational time provided by the VNS is compensated by an increase in the convergence speed. In order to tune these parameters the algorithm ran with a total population of 20 individuals for 10 iterations. All the values were calculated from an average of the 5 best individuals on each instance. Also, to allow comparison in the same plot, all the values are relative to the maximum value in each instance. The instance plots can be read in the blue left vertical axis (corresponding to the total completion time), whereas the computational time plot is read in the right orange vertical axis.

Percentage of best population to perform VNS

Chart 5.6 contains a study (first three instances) for the percentage of the best population to undergo the VNS, while maintaining the number of iterations in the VNS constant and equal to six. The remaining plots for instances 4 to 18 can be consulted in the appendix A.2.1.

After analysing each instance, the percentage of population to undergo VNS that presents the fitter results can be seen in the following table 5.12. Also the number of occurrences on each percentage is presented in table 5.13.

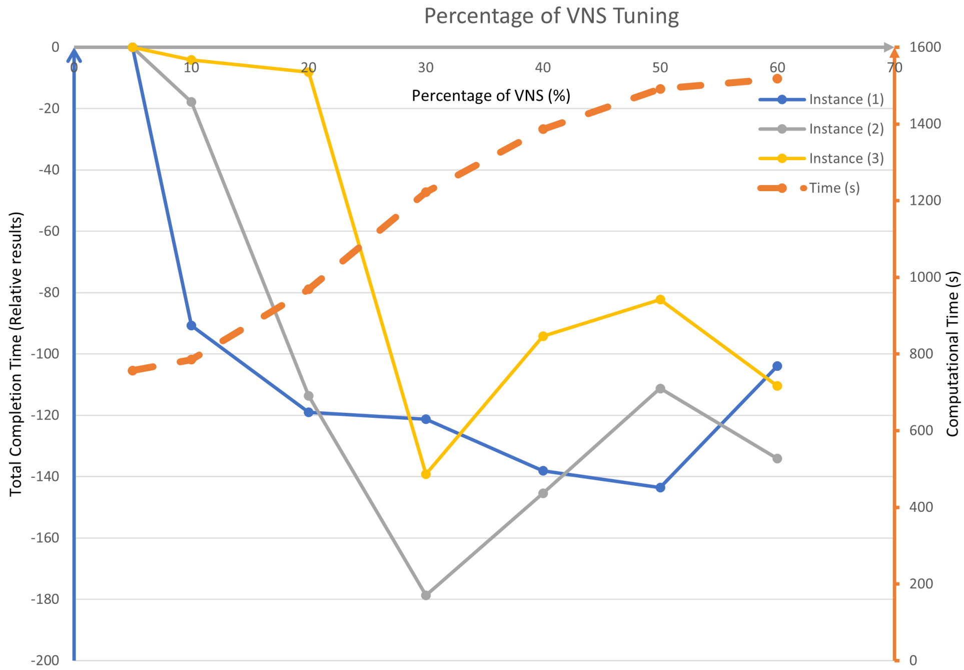


Figure 5.6: Study of the Total Completion Time with different percentage of population accepted for VNS (Instances 1 - 3)

Table 5.12: Best Percentage of VNS in each instance

Instance	Best VNS Percentage (%)
1	20 or 40
2	30
3	30
4	30
5	10, 40 or 60
6	40
7	30
8	10 or 30
9	10
10	20 or 50
11	30
12	30
13	30 or 40
14	20
15	20 or 30
16	20 or 60
17	5
18	20

Table 5.13: Number of Occurrences on each percentage

Best VNS Percentage (%)	**5**	**10**	**20**	**30**	**40**	**50**	**60**
Number of Occurrences	2	3	7	7	4	1	2

The criteria to select the best percentage of VNS was to select the lowest value with the lowest possible amount of computational time. Therefore, for the case in figure 5.6, in instance 1 the 20% mark and the 40% mark were both selected. Table 5.13 indicates that the best percentage of VNS is found between 20 and 30 percent. Therefore, a percentage of 25% for the VNS was chosen.

Number of iterations in the VNS

In order to tune the number of iterations, the same procedure was implemented resulting in the following figure 5.7 for the first 3 instances:

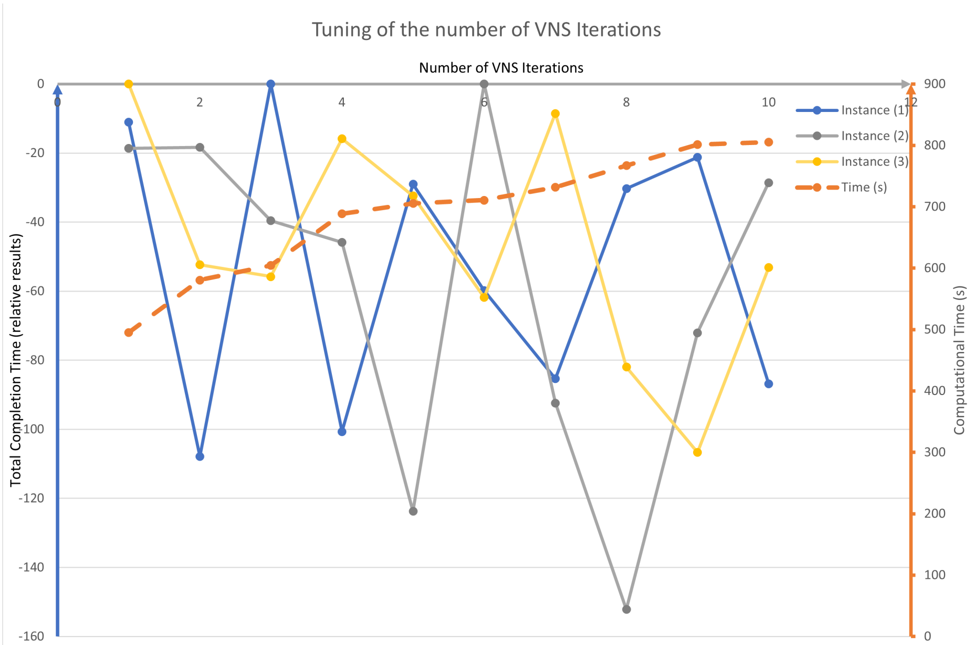


Figure 5.7: Study of the Total Completion Time with different VNS iterations (Instances 1 - 3)

After analysing each instance, the number of iterations that presents the fitter results can be seen in the following table 5.14. Also the number of occurrences on each percentage is presented in table 5.15.

Table 5.14: Best Number of Iterations in VNS in each instance

Instance	Best Number of Iterations (%)
1	2
2	5 or 8
3	3 or 9
4	4 or 6
5	4 or 8
6	5
7	2 or 8
8	5
9	4
10	4
11	6
12	4
13	2
14	4
15	5 or 7
16	6
17	4
18	5

Table 5.15: Number of Occurrences on each iteration

Best Number of VNS Iterations	2	3	4	5	6	7	8
Number of Occurrences	3	1	7	5	3	1	3

The criteria to select the most suitable number of iterations was the same as the one used for the percentage of best population for VNS. Therefore, for the case in figure 5.6, in instance 1, two iterations is the most suitable number. Table 5.15 indicates that most suitable number of iterations is four (with 7 occurrences).

The remaining plots for instances 4 to 18 can be consulted in appendix A.2.2.

5.6 Dynamic Termination Criteria

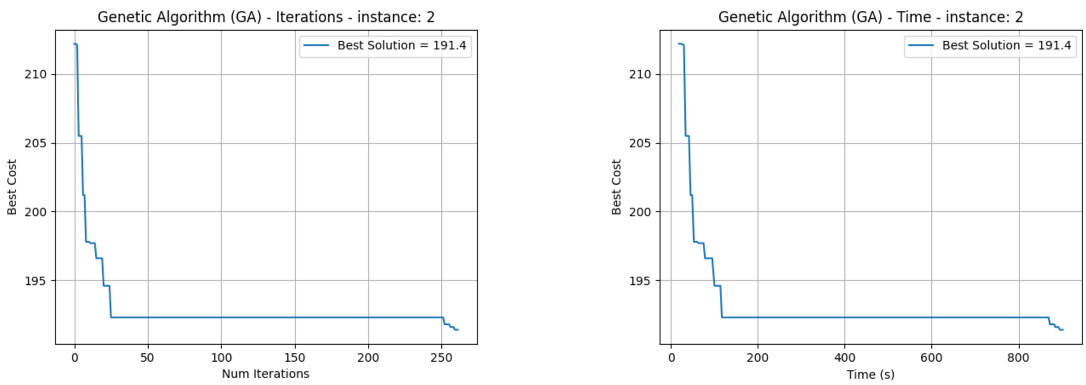
To understand whether the Dynamic Termination Criteria presented fitter results, a study was conducted in the medium instances (10 jobs) since, in these, for a computational time of 15 minutes one can obtain converged results that went through several iterations before reaching the final values. For the smaller instances the algorithm converges too fast not allowing for a proper comparison. For the instances of 70 jobs it takes much more time. The results were preformed with a total population of 400 individuals. The results are presented in 5.16.

Table 5.16: Termination Criteria Comparison

Nº Jobs	Wor-kers	Flex	Rep	Termination Criteria	No Termination Criteria
10	2	0.3	0	91.5	93.7
			1	191.4	193.2
			2	53.9	56
		0.6	0	121.6	122.8
			1	107.9	107.2
			2	130.6	131.7
	3	0.3	0	108.4	108.1
			1	123.1	123.1
			2	96.3	96.4
		0.6	0	60.7	60.8
			1	95.4	96.7
			2	72.9	72.9
	7	0.3	0	103.0	103.4
			1	120.6	120.1
			2	93.1	93.0
		0.6	0	105.1	104.9
			1	99.9	102
			2	89.6	89.6
Total				**1865**	**1876**

From table 5.16, as expected, the Termination Criteria proves fitter results for the total completion time and will therefore be used in the final results.

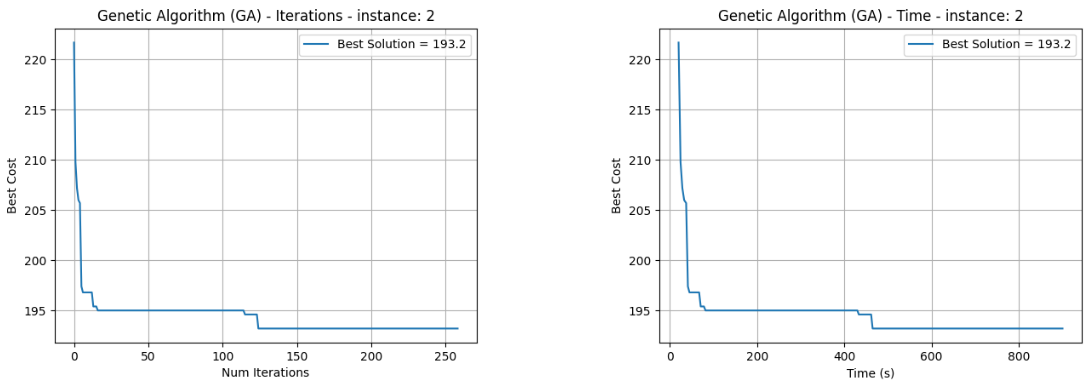
The following figure 5.8 depicts a case where, due to the Dynamic termination Criteria, the algorithm was able to be improved in the last iterations of the study. Also in figure 5.9, the convergence plots of the algorithm while not using the Dynamic Termination Criteria is shown:



(a) Iteration Plot

(b) Time Plot

Figure 5.8: Convergence plot of Instance 2 with Termination Criteria



(a) Iteration Plot

(b) Time Plot

Figure 5.9: Convergence plot of Instance 2 with no Termination Criteria

5.7 Final Results Comparison

The following table 5.18 contains the total completion time (in hours) obtained by the implemented GA with and without VNS, alongside with benchmark results from the work completed by Martins et al. [2]. Highlighted in green and yellow are the best and second best results for each instance, respectively. The parameters used for these results were the ones obtained from the tuning operations (5.2 to 5.5) and can be found in table 5.17.

Table 5.17: Parameters used in the final results

Population Size	400
Initial Population Weights	W1 = 100 | W2 = 0 | W3 = 0
Incremental Initialization?	Yes
Elitism (%)	5
Elite Start	$\frac{1}{4}$ Population size
Percentage of offspring (%)	70
Crossover Weights	C1 = 10 | C2 = 5 | C3 = 85
Mutation Percentage (%)	30
Termination Criteria	250 Iterations or 3h (big instances)

From table 5.18 it is proven that both Genetic Algorithms surpass the Branch and Cut algorithm for large instances with the simple GA surpassing with a margin of 57%. Between both GA's, the one without VNS proves fitter results for medium and large sized instances by a margin of 0.6% and 0.4% respectively. For small instances, the GA with VNS obtains fitter results by 0.1% when compared to the

Table 5.18: Final Results comparison

Nº Jobs	Workers	Flex	Rep	GA	GA + VNS	B&C
5	2	0.3	0	41.3	41.3	41.3
			1	54.9	55.1	54.0
			2	23.6	23.6	23.6
		0.6	0	42.3	41.8	41.7
			1	45.8	45.5	44.3
			2	23.6	23.6	23.3
	3	0.3	0	70.2	70.2	67.7
			1	34.3	34.3	33.3
			2	43.3	43.3	40.6
		0.6	0	27.4	27.4	27.4
			1	26.8	26.8	26.7
			2	33.0	33.0	32.3
	7	0.3	0	29.7	29.7	26.7
			1	46.4	46.4	46.4
			2	24.1	24.1	21.4
		0.6	0	26.1	26.0	26.0
			1	42.1	42.1	42.1
			2	27.0	27.0	27.0
10	2	0.3	0	92.1	91.2	89.8
			1	188.8	187.8	189.5
			2	54.7	55.4	54.4
		0.6	0	120.7	120.1	120.3
			1	108.4	113.3	111.3
			2	130.2	130.1	127.0
	3	0.3	0	108.2	108.2	106.3
			1	123.4	122.5	114.3
			2	96.4	96	94.8
		0.6	0	61.3	60.7	60.2
			1	90.4	92.9	105.2
			2	73.9	75.2	76.0
	7	0.3	0	103	105.1	113.4
			1	119.6	120	119.9
			2	93.1	93.8	87.1
		0.6	0	102.6	106.5	129.9
			1	101.6	98.7	141.2
			2	84.4	85.6	99.9
70	2	0.3	0	3526.9	3656.6	8149.5
			1	5522.8	5544.4	9910.8
			2	3539.5	3509.2	-
		0.6	0	3744	3560.8	10172.7
			1	3888.7	3887.9	11286.7
			2	3906.4	4366.4	9030.0
	3	0.3	0	3304.5	3366.4	8541.5
			1	6401.2	6431.8	-
			2	4077.1	4102.9	8423.6
		0.6	0	2502.6	2466.8	5829.2
			1	2828.1	2776.8	7738.7
			2	1795.3	1827	3573.9
	7	0.3	0	3111.2	3108.9	8255.7
			1	4249.6	4178.9	9559.3
			2	2551.2	2563.4	6130.3
		0.6	0	3772.1	3724	-
			1	1776.6	1811.2	-
			2	3069.7	2926.5	-
Total				**66082.2**	**66334.2**	-

Table 5.20: Encoding table of the final solution for instance 1

Job	2.1	2.2	2.3	3.1	5.1	5.2	4.1	1.1	1.2	1.3	4.2	5.3	3.2	4.3	3.3
Machine	1	6	7	4	4	6	3	7	3	1	5	5	4	7	7
Worker	1	1	2	2	2	1	1	1	1	2	1	1	1	2	1

simple GA. For small instances the Branch and Cut algorithm surpasses both GA's (by a margin of 2.3% to the GA + VNS). For medium instances the BC presents better results when the number of workers is two, but as the instance size increases towards seven workers it looses it's advantage. Therefore, for medium instances the GA performs better than the BC by a margin of 0.6% and the GA + VNS obtains the same overall results as the BC. The following table 5.19 outlines these comparisons:

Table 5.19: Comparison between algorithms for the 3 types of Instances

Instance Type	Best Algorithm	Second Best Algorithm	Worst Algorithm
Small (5 Jobs)	B&C	GA + VNS	GA
Medium (10 Jobs)	GA	GA + VNS & B&C	-
Large (70 Jobs)	GA	GA + VNS	B&C

Representation of the Solution

In order to represent the solution a Gantt Chart for both workers and machines is computed. For the solution of the first instance in table 5.18 with a total completion time of 41.3 hours (and a makespan of 12.6 hours), it's Gantt Charts are represented in figures 5.10 and 5.11, to which corresponds the encoding table 5.20.

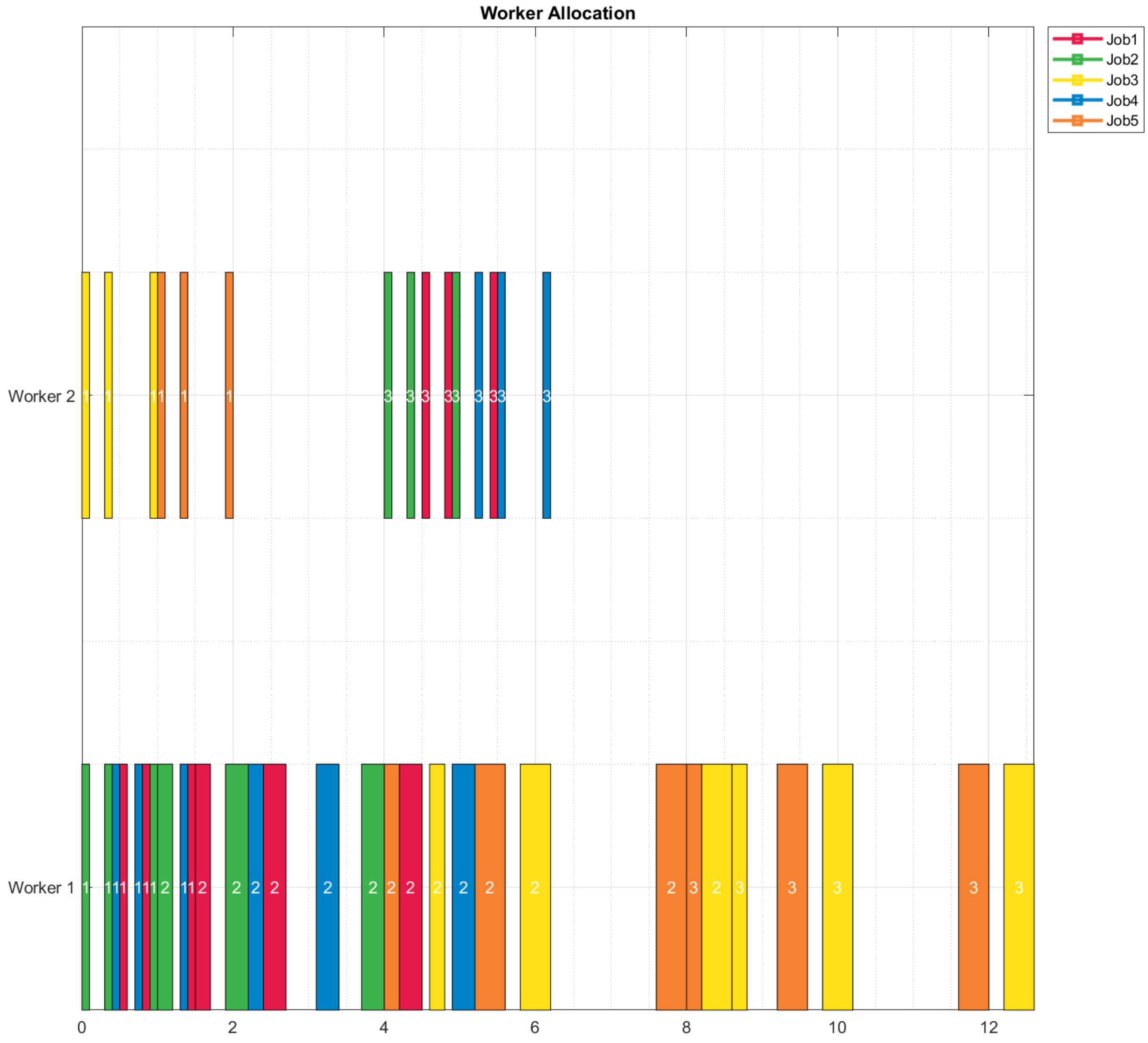


Figure 5.10: Worker Gantt Chart of first instance of the small instances

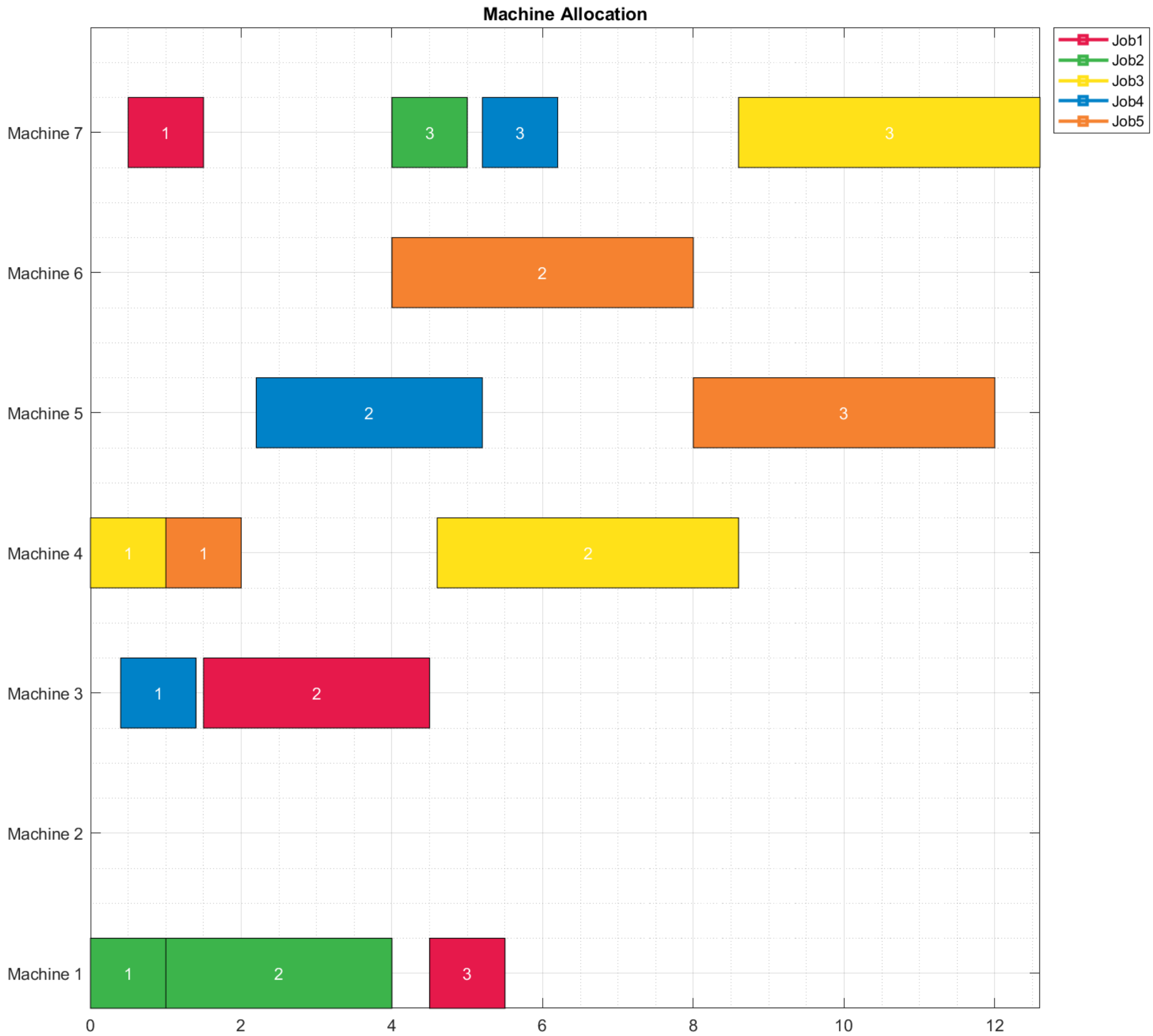


Figure 5.11: Machine Gantt Chart of first instance of the small instances

6

Conclusions

Contents

6.1 Conclusions

The objective of this thesis was to develop tools capable of creating a real life-size schedule for a Quality Control laboratory environment using a meta-heuristic algorithm. To achieve this objective, a hybrid Genetic algorithm was implemented to minimize the total completion time. This algorithm was designed specifically for the problem at hand and introduced novel approaches to former studies in literature, even when considering to the more general manufacturing scheduling. Compatibility tables were successfully used in the context of laboratory scheduling to inform upon the availability of resources (machines and workers).

The Initial Population had a system of weights which corresponded to probabilities of performing a certain path in the assignment of the genes. The random Initialization (w_1) proved the best results overall. Additionally, the prohibition conditions proved fitter results regardless the value used in each of them. The fitter results were obtained while restricting the number of workers in a row to 4 and machines to 3. The Incremental Initial Population with worker selection also proved to improve the algorithm's convergence and overall results. Regarding the Crossover, unlike the initial population, the fitter results in this phase were found with a combination of the three weights: $c_1 = 10$, $c_2 = 5$ and $c_3 = 85$. The general GA parameters were studied and a tuning was performed for the total population size, the total percentage of offspring and percentage of mutation. The Variable Neighborhood Search proved to be best when the percentage of the best percentage of population chosen to perform VNS was 25% and when a total of four iterations were performed.

The GA with VNS proved fitter results when compared to the simple GA only for the small instances with five jobs but could not match with the optimal values from the Branch and Cut algorithm. The final results and comparison with bench mark tests was performed and the implemented algorithm proved fitter results for the GA without Variable Neighborhood search for large and medium instances. Since the objective of this dissertation was to create a real life-size schedule for a Quality Control laboratory, the 70 job instances are the ones mimicking this dimension. The Genetic Algorithm (GA) in this study improved the results from previous studies by a total of 57%. Therefore, the objectives of this dissertation were met and a novel implementation was successfully introduced.

6.2 Future Work

The scheduling of resources extends further than the dual resource constrained flexible job shop problem. As introduced in the literature review in chapter 1.2.5, there are other factors that can be considered. For example, workers do not all perform with the same level of productivity, resource constraints are often not limited to workers and machines. Furthermore, in some cases it may be useful to consider other objective functions and optimization criteria, even simultaneously. Therefore, as for future work, this thesis could be extended to a multi-resource or multi-objective scheduling problem. It is worth mentioning that the algorithm is also prepared to minimize the makespan which can be particularly useful if one decides to migrate to multi-objective scheduling. Heterogeneous workers may also be considered in future work and learning curves implemented in the study.

Due to unpredictable events, there may be shortage of supply or an urgent order in need to be attended. Therefore, dynamic scheduling is often of high importance in scheduling problems which could be a valuable extension to this thesis.

Furthermore, the optimal tuning of parameters in a genetic algorithm is often complicated to achieve since parameters are not mutually exclusive and often depend on each other. A further study could be performed in order to understand how these parameters could be better tuned and access which parameters have higher correlation between each other. Perhaps a cross-tuning of the parameters could be studied in future work, or a tuning using Bayesian optimization.

The study of the quality control laboratory scheduling is still very recent. This is known to be the first study in this environment to make use of a meta-heuristic algorithm. Further meta-heuristics could be developed and compared to the one in this work and possibly find which meta-heuristic provides fitter results to the problem at hand.

Also, it could be of interest to study how the current Genetic Algorithm could perform faster and more efficient. As an example, parallel computing could be explored in the algorithm in order to reduce the fitness evaluation phase computational time.

Bibliography

[1] D. Wolejszo, "History of project scheduling," https://www.linkedin.com/pulse/history-project-scheduling-dariusz-wolejszo/, 2019.

[2] M. Miguel, L. Joaquim, V. Tiago, C. Bernardo, C. Andrea, M. V. Susana, and S. João, "Minimizing total completion time in large-sized pharmaceutical quality control scheduling," *pending magazine approval*.

[3] P. Weaver, "A brief history of scheduling: Back to the future," *myPrimavera Conference*, no. April, p. 24.

[4] M. B. Horta Mesquita Da Cunha, "Scheduling of Flexible Job Shop Problem in Dynamic Environment," pp. 1–10, 2017.

[5] M. Christoph Frey, *Production Management A, course book*, 2020.

[6] A. Costigliola, F. A. Ataide, S. M. Vieira, and J. M. Sousa, "Simulation Model of a Quality Control Laboratory in Pharmaceutical Industry," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 9014–9019, 2017.

[7] Statista, "Annual growth of pharmaceutical rd spending in europe and the u.s. between 2006 and 2020," https://www.statista.com/statistics/315959/annual-growth-rate-of-pharmaceutical-research-and-development-expenditure/, 2021.

[8] FDA, "Pharmaceutical quality control labs," https://www.fda.gov/inspections-compliance-enforcement-and-criminal-investigations/inspection-guides/pharmaceutical-quality-control-labs-793, 2014.

[9] M. Colledani and T. Tolio, "Joint design of quality and production control in manufacturing systems," *CIRP Journal of Manufacturing Science and Technology*, no. 3, pp. 281–289.

[10] Brucker and Schlie, "Job-shop scheduling with multi-purpose machines," Computing 45 (4): 369–375 (1991).

[11] Dauzère-Pérès and Paulli, “An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search,” Annals of Operations Research 70: 281–306 (1997).

[12] F. Pezzella, G. Morganti, and G. Ciaschetti, “A genetic algorithm for the Flexible Job-shop Scheduling Problem,” *Computers and Operations Research*, vol. 35, no. 10, pp. 3202–3212, 2008.

[13] I. Driss, K. N. Mouss, and A. Laggoun, “A new genetic algorithm for flexible job-shop scheduling problems,” *Journal of Mechanical Science and Technology*, vol. 29, no. 3, pp. 1273–1281, 2015.

[14] M. Dhiflaoui, H. E. Nouri, and O. B. Driss, “ScienceDirect ScienceDirect ScienceDirect Dual-Resource Constraints in Classical and Flexible Dual-Resource Constraints in Classical and Flexible Job Shop Problems : A State-of-the-Art Review Job Shop Problems :,” *Procedia Computer Science*, pp. 1507–1515.

[15] H. V. Kher, “Examination of worker assignment and dispatching rules for managing vital customer priorities in dual resource constrained job shop environments,” Computers Operations Research 27 525-537 (2000).

[16] V. Patel, “Scheduling in a dual resource constrained system using genetic algorithms,” p. 108, 1997.

[17] H. Eimaraghy, I. Manufacturing, and S. Ims, “Scheduling of Manufacturing Systems Under Dual-Resource Constraints Using Genetic Algorithms,” vol. 19, no. 3, pp. 186–201, 2000.

[18] J. T. Felan and T. D. Fry, “Multi-level heterogeneous worker flexibility in a Dual Resource Constrained (DRC) job-shop,” *International Journal of Production Research*, no. 14, pp. 3041–3059, jan.

[19] H. Yue, J. Slomp, E. Molleman, and D. J. Van Der Zee, “Worker flexibility in a parallel dual resource constrained job shop,” *International Journal of Production Research*, vol. 46, no. 2, pp. 451–467, 2008.

[20] C. Xianzhou and Y. Zhenhe, “An improved genetic algorithm for dual-resource constrained flexible job shop scheduling,” *Proceedings - 4th International Conference on Intelligent Computation Technology and Automation, ICICTA 2011*, vol. 1, pp. 42–45, 2011.

[21] D. Lei and X. Guo, “Variable neighbourhood search for dual-resource constrained flexible job shop scheduling,” *International Journal of Production Research*, vol. 52, no. 9, pp. 2519–2529, 2014.

[22] M. Yazdani, M. Zandieh, R. Tavakkoli-moghaddam, and F. Jolai, “Two meta-heuristic algorithms for the dual-resource constrained exible job-shop scheduling problem,” vol. 22, pp. 1242–1257, 2015.

[23] A. B. Paksi and A. Ma'Ruf, "Flexible Job-Shop Scheduling with Dual-Resource Constraints to Minimize Tardiness Using Genetic Algorithm," *IOP Conference Series: Materials Science and Engineering*, vol. 114, no. 1, 2016.

[24] J. Zhang, W. Wang, and X. Xu, "A hybrid discrete particle swarm optimization for dual-resource constrained job shop scheduling with resource flexibility," *Journal of Intelligent Manufacturing*, vol. 28, no. 8, pp. 1961–1972, 2017.

[25] Q. Zhong and H. Yang, "Optimization algorithm simulation for dual-resource constrained job-shop scheduling," *International Journal of Simulation Modelling*, vol. 17, no. 1, pp. 147–158, 2018.

[26] M. G. Helander, "Seven common reasons to not implement ergonomics," nternational Journal of Industrial Ergonomics 25 (1): 97–101 (2000).

[27] M. Lang and H. Li, "Research on dual-resource multi-objective flexible job shop scheduling under uncertainty," *2011 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce, AIMSEC 2011 - Proceedings*, no. 2009, pp. 1375–1378, 2011.

[28] K. Z. Gao, P. N. Suganthan, Q. K. Pan, M. F. Tasgetiren, and A. Sadollah, "Artificial bee colony algorithm for scheduling and rescheduling fuzzy flexible job shop problem with new job insertion," *Knowledge-Based Systems*, pp. 1–16.

[29] Ö. U. Araz and L. Salum, "A multi-criteria adaptive control scheme based on neural networks and fuzzy inference for DRC manufacturing systems," vol. 7543, 2010.

[30] J. L. Andrade-pineda, D. Canca, and P. G. Calle, "Scheduling a dual-resource flexible job shop with makespan and due date-related criteria," *Annals of Operations Research*, no. 1, pp. 5–35.

[31] X.-l. L. Zheng and L. Wang, "A knowledge-guided fruit fly optimization algorithm for dual resource constrained flexible job-shop scheduling problem," *International Journal of Production Research*, no. 18, pp. 5554–5566.

[32] R. Wu, Y. Li, S. Guo, and W. Xu, "Solving the dual-resource constrained flexible job shop scheduling problem with learning effect by a hybrid genetic algorithm," vol. 10, no. 10, pp. 1–14, 2018.

[33] X. Wu, J. Peng, X. Xiao, and S. Wu, "An effective approach for the dual-resource flexible job shop scheduling problem considering loading and unloading," *Journal of Intelligent Manufacturing*, no. 3, pp. 707–728.

[34] X. Gong, Q. Deng, G. Gong, W. Liu, and Q. Ren, "A memetic algorithm for multi-objective flexible job-shop problem with worker flexibility," *International Journal of Production Research*, no. 7, pp. 2506–2522.

[35] G. Gong, Q. Deng, X. Gong, W. Liu, and Q. Ren, “A new double flexible job-shop scheduling problem integrating processing time , green production , and human factor indicators,” *Journal of Cleaner Production*, pp. 560–576.

[36] M. Yazdani and M. Zandieh, “Evolutionary algorithms for multi-objective dual-resource constrained flexible job-shop scheduling problem,” *OPSEARCH*, no. 3, pp. 983–1006.

[37] L. Gao and Q. K. Pan, “A shuffled multi-swarm micro-migrating birds optimizer for a multi-resource-constrained flexible job shop scheduling problem,” *Information Sciences*, pp. 655–676.

[38] M. Cunha, L. Viegas, M. Cunha, M. Vieira, M. T, and M. Susana, “Dual Resource Constrained Scheduling - Quality Control Laboratories,” *IFAC PapersOnLine*, no. 13, pp. 1421–1426.

[39] M. Akbar and T. Irohara, “Engineering Applications of Artificial Intelligence Metaheuristics for the multi-task simultaneous supervision dual resource-constrained scheduling problem,” *Engineering Applications of Artificial Intelligence*, no. March, p. 104004.

[40] ——, “A social-conscious scheduling model of dual resources constrained identical parallel machine to minimize tardiness and maximize operator productivity,” *Proceedings of International Conference on Computers and Industrial Engineering, CIE*, 2018.

[41] Z. Jinghui, H. Xiaomin, G. Min, and Z. Jun, “Comparison of performance between different selection strategies on simple genetic algorithms,” *Proceedings - International Conference on Computational Intelligence for Modelling, Control and Automation, CIMCA 2005 and International Conference on Intelligent Agents, Web Technologies and Internet*, vol. 2, pp. 1115–1120, 2005.

A

Tuning

A.1 Initial Results - No Tuning

The first results for the Genetic Algorithm A.1 were obtained with the simpler approaches where only the one point crossover was implemented, no initial population nor crossover calibration and no VNS. A stopping criteria was implemented that fires if after 30 iterations the algorithm has no improvements. Also, the algorithm is limited to run for a maximum of 15 minutes regardless of the number of iterations it performs.

Table A.1: Initial Results - Not Tuned

Nº Jobs	Workers	Flex	Rep	GA	Time (s)	Pop	Num It	Nº of last non improving it:
5	2	0.3	0	41.7	12.3	150	40	30
			1	55.1	12.7	150	42	30
			2	23.6	6.8	150	35	30
		0.6	0	42.6	13.6	150	43	30
			1	45.9	12.7	150	42	30
			2	23.6	5.5	150	31	30
	3	0.3	0	70.2	12.7	150	35	30
			1	34.3	10.1	150	39	30
			2	43.3	9.4	150	33	30
		0.6	0	27.4	7.7	150	38	30
			1	26.8	8.7	150	40	30
			2	33	10.2	150	39	30
	7	0.3	0	29.7	6.6	150	32	30
			1	46.4	9.6	150	32	30
			2	24.1	8.2	150	39	30
		0.6	0	26.1	8.2	150	40	30
			1	42.1	10.6	150	39	30
			2	27	6.8	150	30	30
10	2	0.3	0	91.6	32.2	150	64	30
			1	192.4	70.1	150	83	30
			2	56.7	27.2	150	68	30
		0.6	0	121.3	56.7	150	64	30
			1	108	56.8	150	67	30
			2	130.9	50.3	150	57	30
	3	0.3	0	108.4	28.1	150	42	30
			1	123.8	47.1	150	57	30
			2	96.5	25.4	150	43	30
		0.6	0	63	23.8	150	59	30
			1	92.9	25.6	150	42	30
			2	76.8	32.5	150	68	30
	7	0.3	0	103.9	23.3	150	37	30
			1	124.3	22.8	150	39	30
			2	95.3	20.2	150	36	30
		0.6	0	111.2	45.8	150	71	30
			1	101.1	42.6	150	67	30
			2	88.6	35.3	150	69	30
70	2	0.3	0	3912.1	907.4	150	72	2
			1	5957.2	903.3	150	38	0
			2	3671.7	905.9	150	101	0
		0.6	0	4152.7	904.5	150	52	1
			1	4462.6	903.2	150	53	2
			2	4501.6	911.7	150	53	0
	3	0.3	0	3446.3	907.2	150	68	3
			1	7094.6	907.6	150	44	0
			2	4414.3	911.6	150	79	1
		0.6	0	2766	906.8	150	92	1
			1	3022.9	900.8	150	88	11
			2	1893.5	900.6	150	138	1
	7	0.3	0	3205	905.9	150	77	2
			1	4507.3	921.3	150	54	1
			2	2861.9	903.7	150	96	6
		0.6	0	4301.5	907.7	150	55	1
			1	2039.1	901.2	150	66	2
			2	3325.4	914.4	150	41	4
Total				**72085.3**				

It can be noticed that for the smaller instances of 5 and 10 jobs the algorithm reaches some kind of optimum. For the largest instance of 70 jobs the algorithm is far from converging into one.

A.2 VNS Tuning

A.2.1 Percentage of best Population

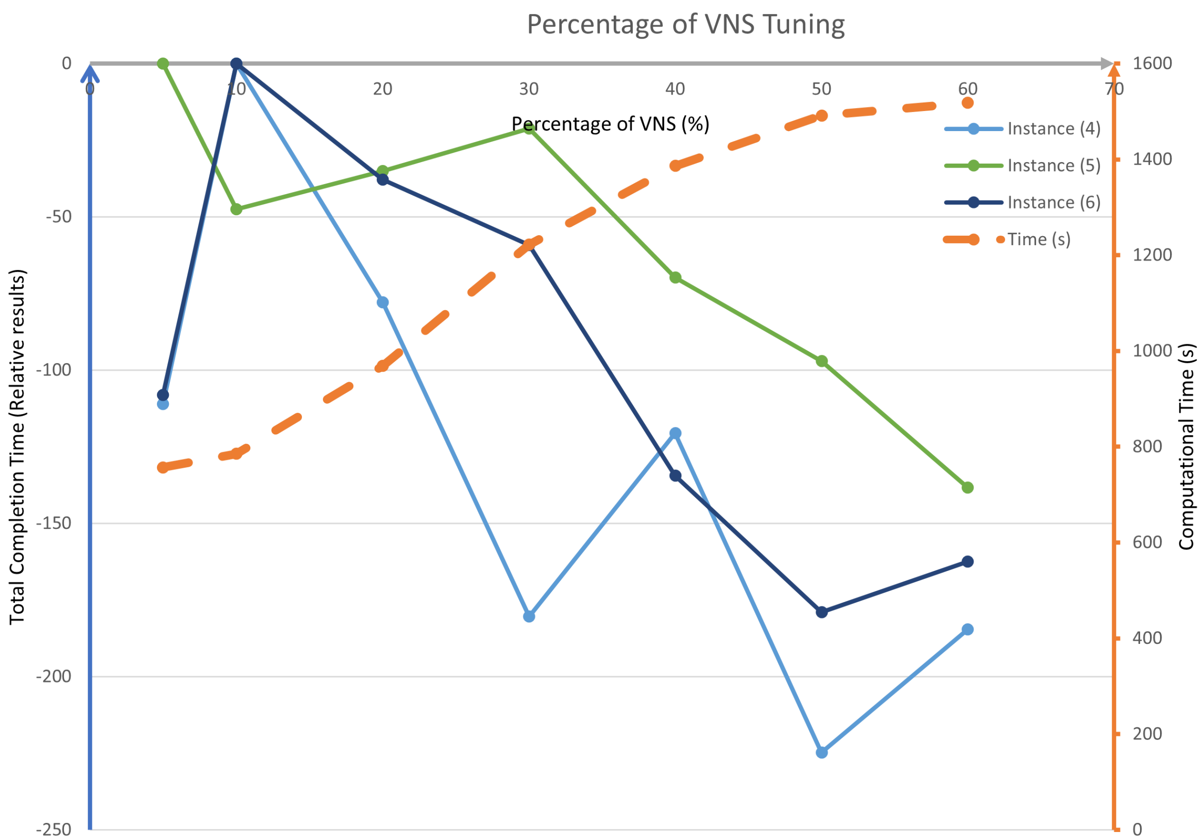


Figure A.1: VNS Tuning Instances 4 - 6

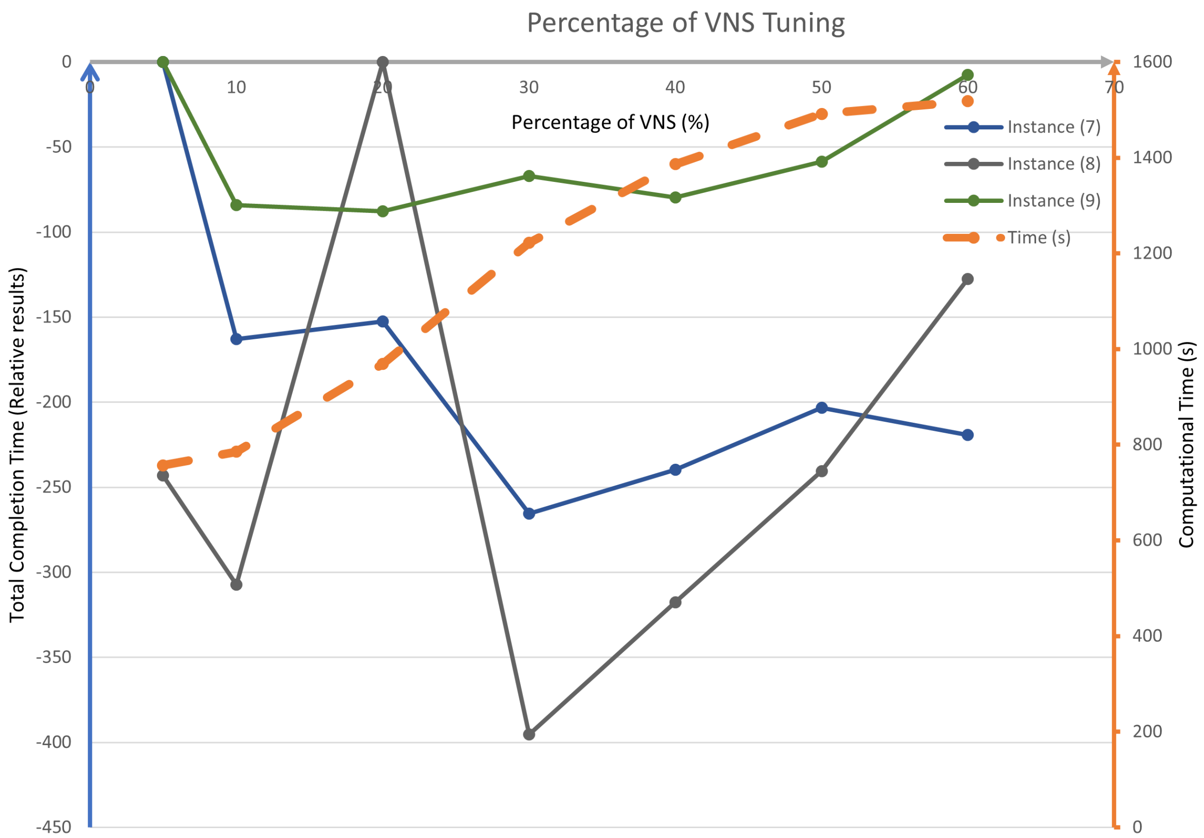


Figure A.2: VNS Tuning Instances 7 - 9

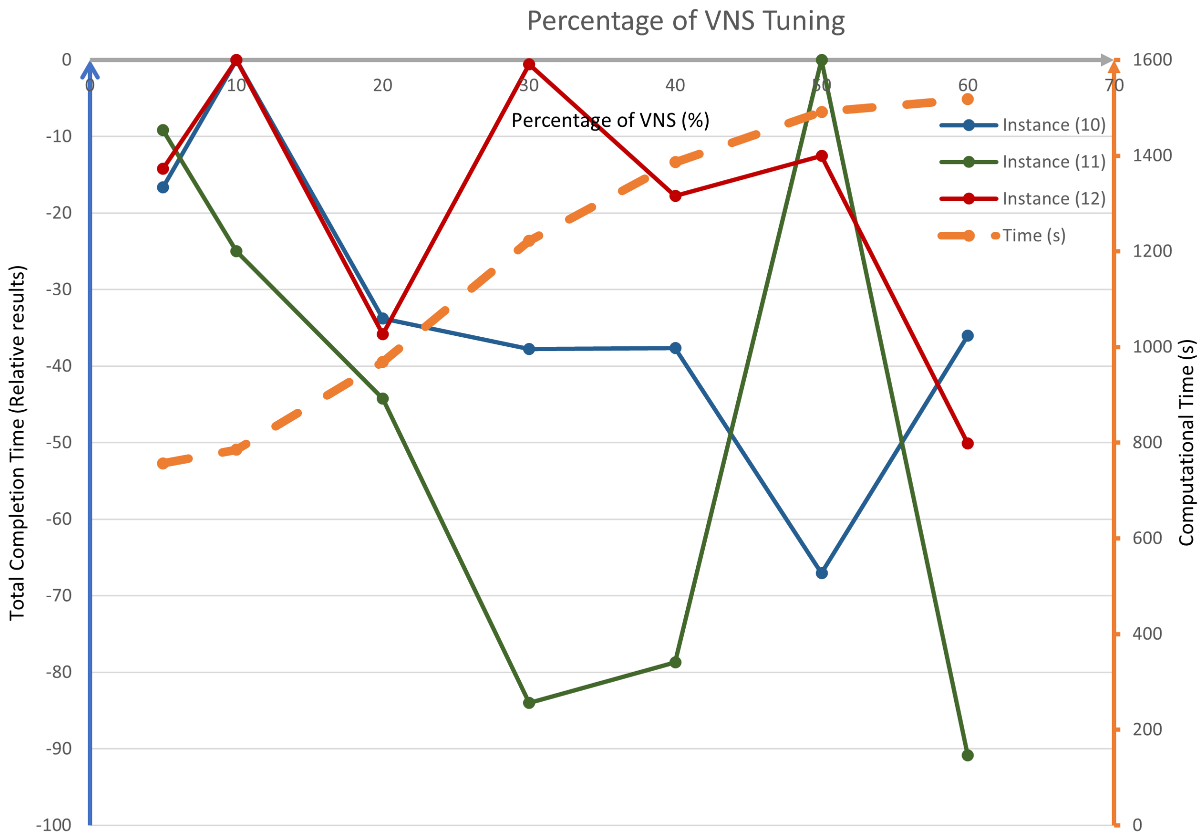


Figure A.3: VNS Tuning Instances 10 - 12

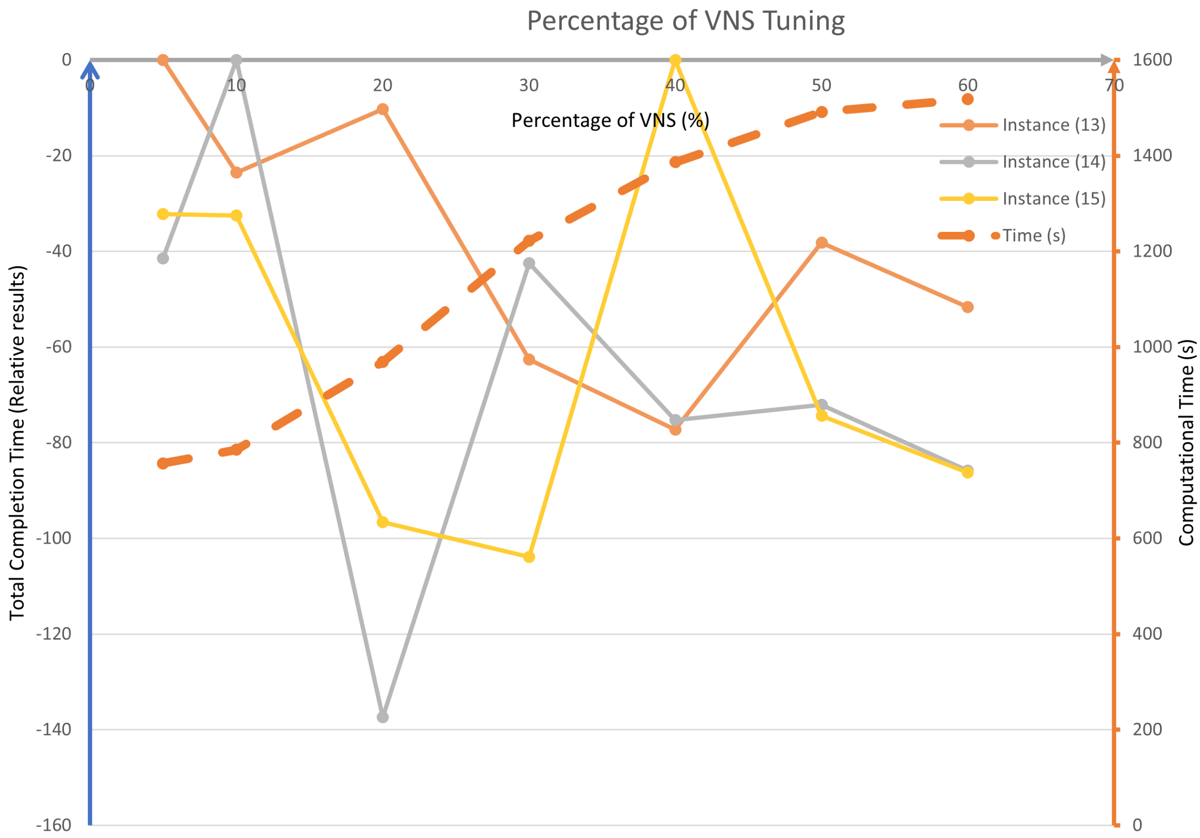


Figure A.4: VNS Tuning Instances 13 - 15

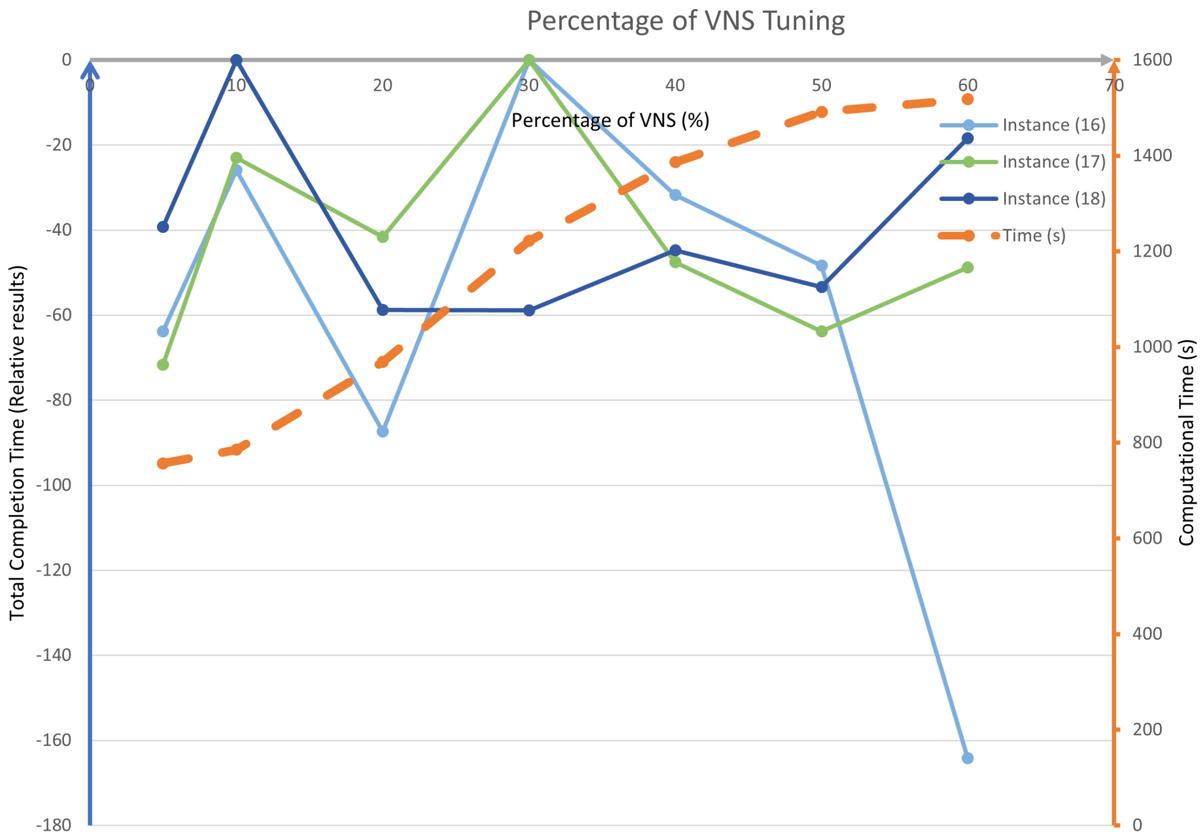


Figure A.5: VNS Tuning Instances 16 - 18

A.2.2 Number of Iterations

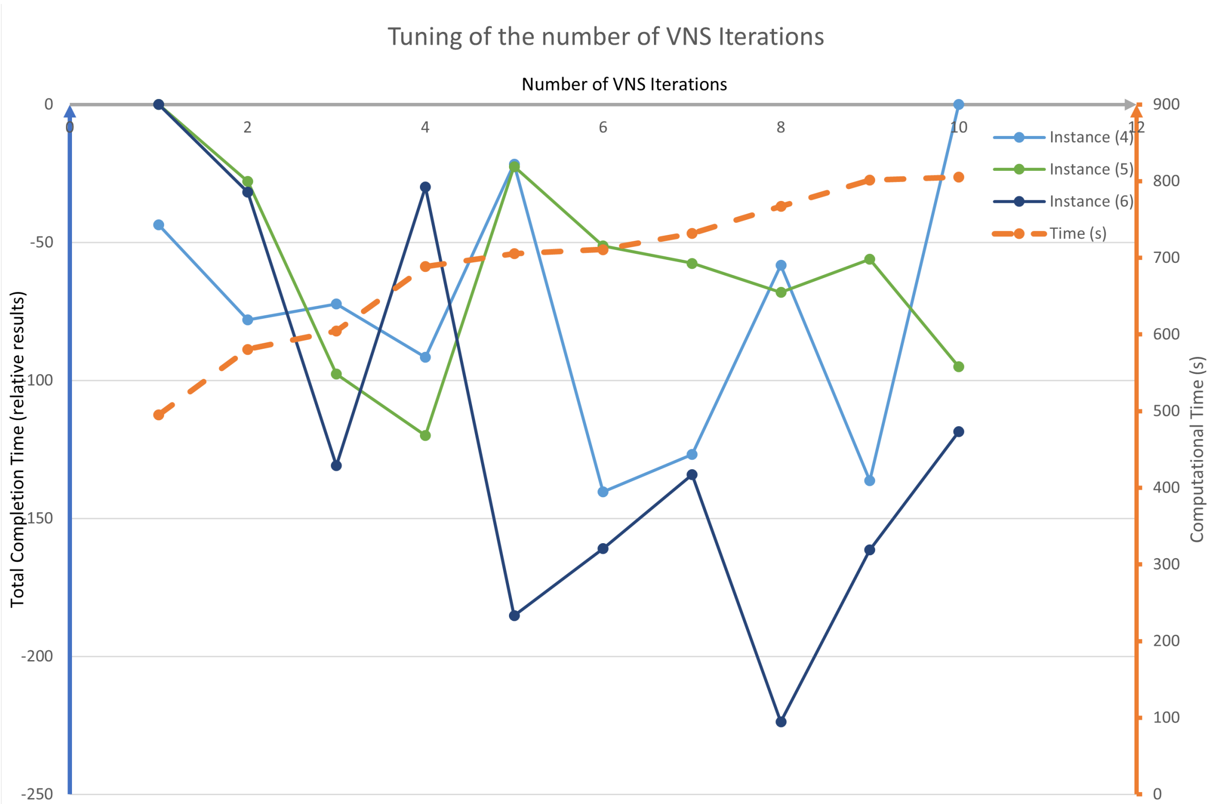


Figure A.6: VNS Iteration Tuning Instances 4 - 6

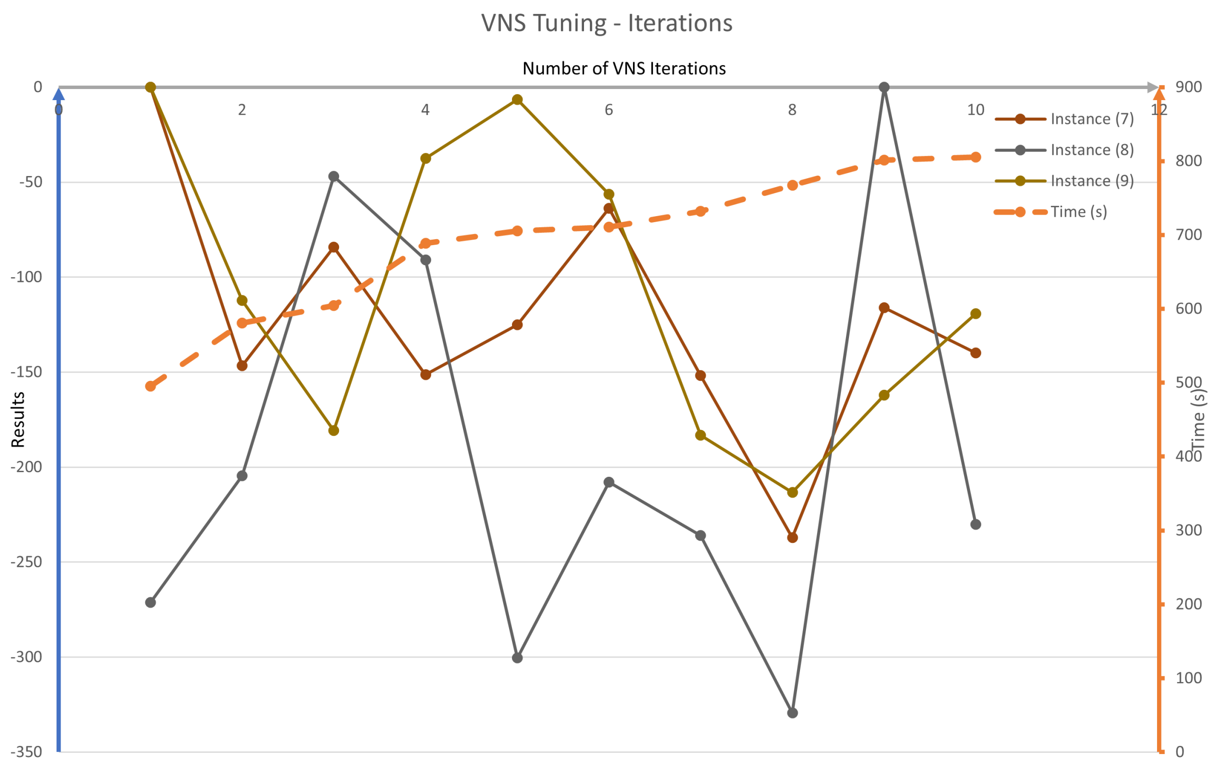


Figure A.7: VNS Tuning Instances 7 - 9

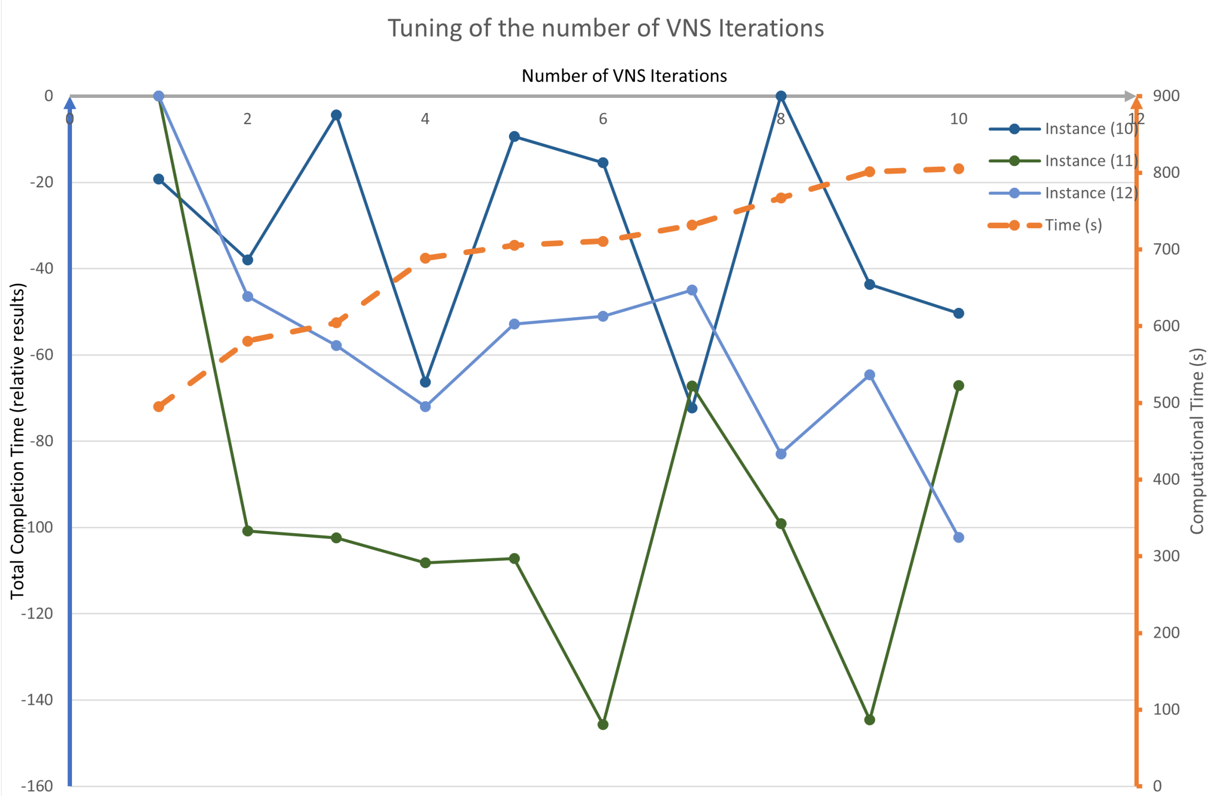


Figure A.8: VNS Iteration Tuning Instances 10 - 12

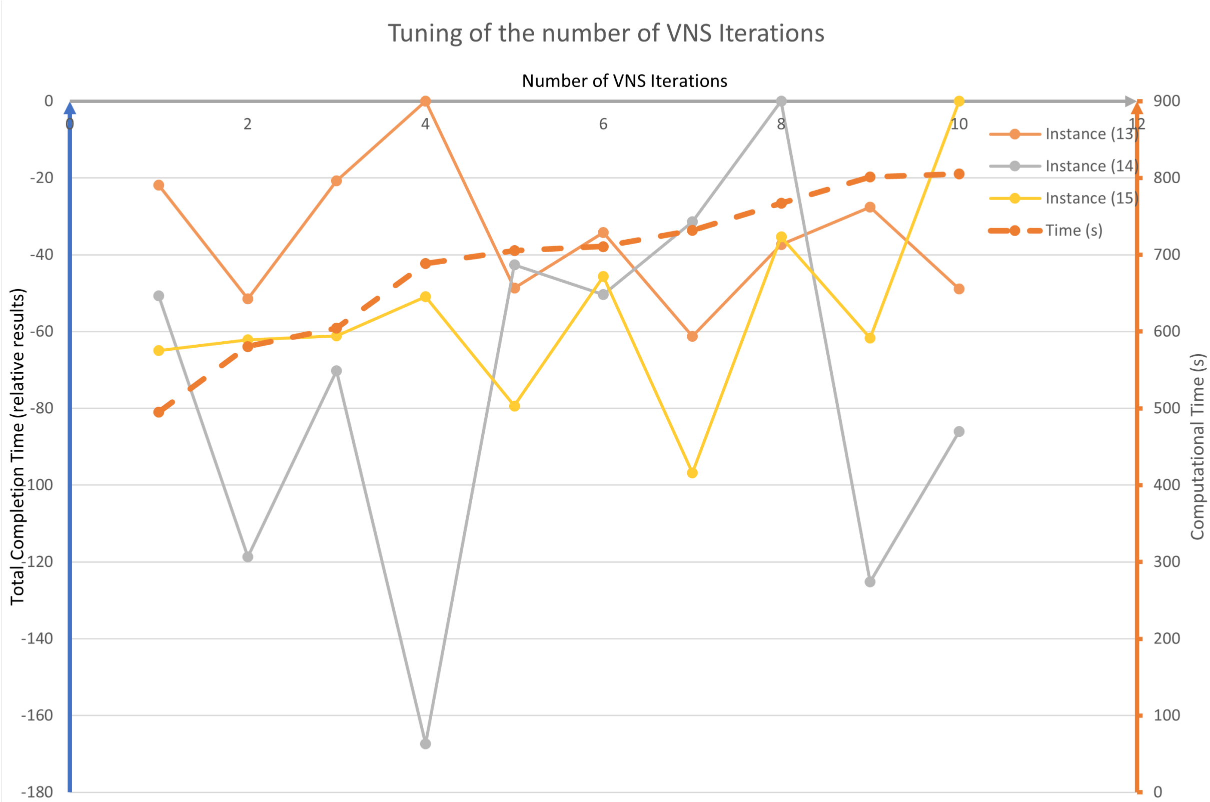


Figure A.9: VNS Iteration Tuning Instances 13 - 15

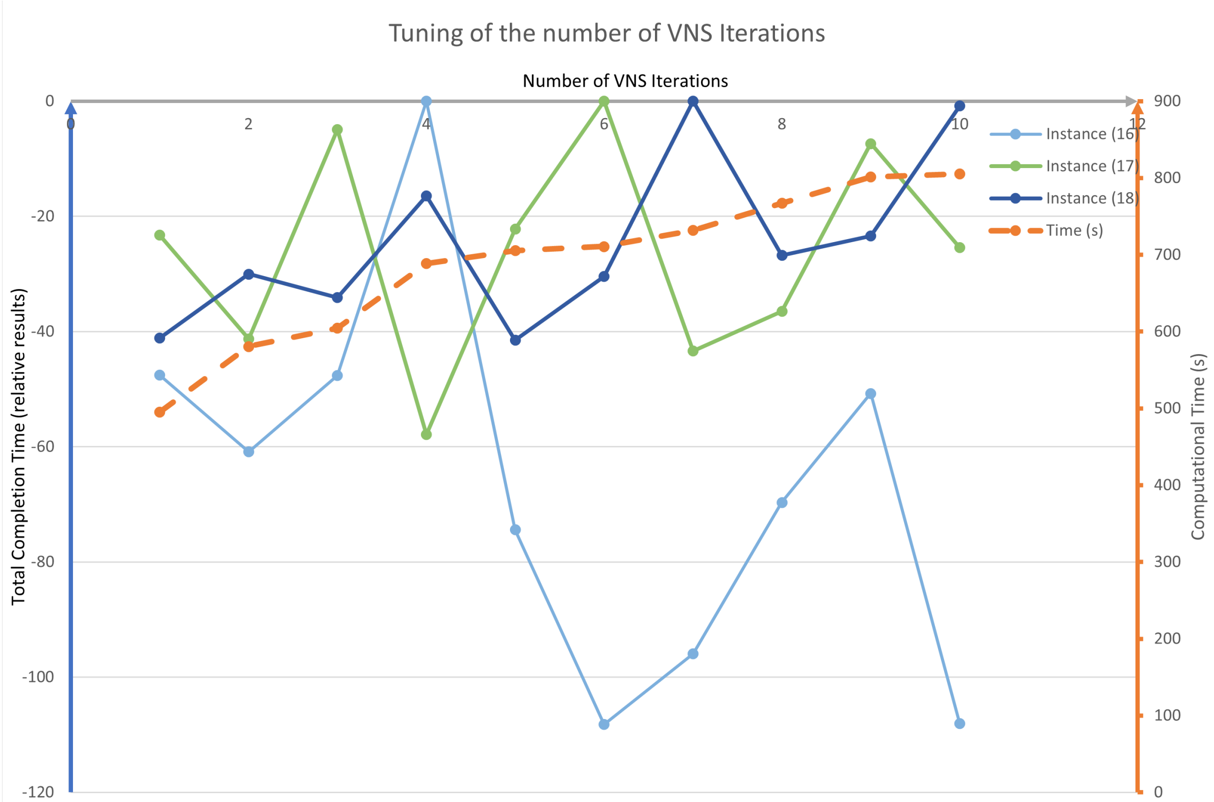


Figure A.10: VNS Iteration Tuning Instances 16 - 18

B

Prohibition Condition

Table B.1: Prohibition Condition values with counting part 1

Nº Jobs	Workers	Flex	Rep	GA W = 3 M = 3	Worker Count	Machine Count	Total Count	GA W = 3 M = 2	Worker Count	Machine Count	Total Count	GA W = 3 M = 4	Worker Count	Machine Count	Total Count	GA W = 3 M = 6	Worker Count	Machine Count	Total Count
70	2	0.3	0	**5001**	5118	1140	6258	**4985**	5123	4049	9172	**5007**	5113	446	5559	**5005**	5140	52	5192
			1	**7356**	0	1271	1271	**7316**	0	5666	5666	**7384**	0	367	367	**7378**	0	27	27
			2	**4544**	0	816	816	**4554**	0	3419	3419	**4541**	0	221	221	**4543**	0	16	16
		0.6	0	**5134**	5537	1029	6566	**5138**	5567	6676	12243	**5134**	5615	177	5792	**5149**	5450	4	5454
			1	**5455**	9681	1496	11177	**5441**	9609	8936	18545	**5449**	9684	286	9970	**5450**	9645	8	9653
			2	**5648**	2341	1507	3848	**5642**	2513	9645	12158	**5650**	2525	279	2804	**5648**	2399	11	2410
	3	0.3	0	**4548**	0	1858	1858	**4521**	0	6476	6476	**4593**	0	613	613	**4585**	0	111	111
			1	**12497**	5087	4804	9891	**12096**	5026	10535	15561	**12783**	5125	2485	7610	**12980**	4997	688	5685
			2	**6172**	1252	1196	2448	**6134**	1253	4843	6096	**6198**	1266	345	1611	**6204**	1136	28	1164
		0.6	0	**3679**	4206	1452	5658	**3647**	4250	8783	13033	**3673**	4183	295	4478	**3674**	4093	8	4101
			1	**3854**	2204	1158	3362	**3850**	2118	7216	9334	**3876**	2133	208	2341	**3870**	2106	6	2112
			2	**2701**	2678	1398	4076	**2686**	2705	7268	9973	**2704**	2668	266	2934	**2729**	2642	14	2656
	7	0.3	0	**4343**	888	810	1698	**4310**	873	3805	4678	**4369**	895	200	1095	**4374**	842	23	865
			1	**6606**	2386	1389	3775	**6502**	2404	6594	8998	**6609**	2532	330	2862	**6598**	2487	30	2517
			2	**3585**	1219	803	2022	**3565**	1300	3533	4833	**3615**	1217	226	1443	**3610**	1233	18	1251
		0.6	0	**5628**	1435	2007	3442	**5558**	1518	10870	12388	**5644**	1411	433	1844	**5667**	1387	16	1403
			1	**2836**	2122	1441	3563	**2816**	2190	8727	10917	**2839**	2181	234	2415	**2831**	2128	10	2138
			2	**4538**	1688	1683	3371	**4512**	1620	9628	11248	**4529**	1600	317	1917	**4553**	1622	15	1637
Average				**5229**				**5182**				**5255**				**5269**			

Table B.2: Prohibition Condition values tuning part 2

Nº Jobs	Workers	Flex	Rep	GA W = 4 M = 3	Worker Count	Machine Count	Total Count	GA W = 4 M = 4	Worker Count	Machine Count	Total Count	GA W = 4 M = 5	Worker Count	Machine Count	Total Count	GA W = 5 M = 5	Worker Count	Machine Count	Total Count
70	2	0.3	0	**5067**	4012	1159	5171	**5056**	4052	400	4452	**5057**	3918	146	4064	**5112**	3226	170	3396
			1	**7378**	0	1343	1343	**7381**	0	332	332	**7384**	0	98	98	**7392**	0	99	99
			2	**4537**	0	819	819	**4546**	0	189	189	**4548**	0	65	65	**4547**	0	54	54
		0.6	0	**5141**	3134	1054	4188	**5149**	3062	194	3256	**5143**	3131	34	3165	**5163**	1846	33	1879
			1	**5533**	5506	1447	6953	**5529**	5440	277	5717	**5529**	5362	41	5403	**5552**	3202	39	3241
			2	**5637**	1289	1548	2837	**5665**	1317	263	1580	**5643**	1281	55	1336	**5661**	652	53	705
	3	0.3	0	**4546**	0	1884	1884	**4568**	0	669	669	**4570**	0	230	230	**4594**	0	282	282
			1	**12552**	2447	4831	7278	**12720**	2456	2435	4891	**12931**	2399	1343	3742	**12905**	1293	1368	2661
			2	**6172**	543	1244	1787	**6216**	492	335	827	**6206**	501	93	594	**6201**	212	100	312
		0.6	0	**3676**	1431	1490	2921	**3683**	1403	288	1691	**3680**	1432	56	1488	**3680**	541	53	594
			1	**3871**	832	1218	2050	**3863**	818	192	1010	**3862**	860	32	892	**3875**	380	28	408
			2	**2703**	937	1449	2386	**2717**	954	302	1256	**2733**	971	60	1031	**2710**	334	86	420
	7	0.3	0	**4358**	251	815	1066	**4387**	255	235	490	**4359**	286	80	366	**4364**	79	67	146
			1	**6597**	738	1422	2160	**6579**	668	317	985	**6610**	764	92	856	**6594**	186	83	269
			2	**3596**	296	811	1107	**3599**	299	198	497	**3627**	295	49	344	**3625**	78	50	128
		0.6	0	**5629**	262	1937	2199	**5663**	260	379	639	**5660**	262	80	342	**5653**	47	70	117
			1	**2837**	554	1429	1983	**2847**	542	295	837	**2855**	548	38	586	**2846**	150	46	196
			2	**4532**	284	1682	1966	**4520**	316	357	673	**4552**	304	62	366	**4543**	58	61	119
Average				**5242**				**5260**				**5275**				**5279**			