Dual Resource Constrained Flexible Job Shop Problem using Hybrid Genetic Algorithm

Application to Quality Control Laboratory Scheduling

Diogo Marta
diogo.marta@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal

December 2021

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.

1. Introduction

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 providing order to businesses. The constant evolution of the scheduling science allows systems to be more complex, with a higher number of activities and sequencing of operations [4]. A particular industry that requires fast and reliable scheduling methods 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 [3]. More specifically, the Quality Control of goods 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 obtained. 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 by promptly detecting deviations in the quality of a manufacturing stream [4].

In literature, researches have been focusing mainly on manufacturing scheduling problems which fail to include the peculiarities of the Quality Control laboratory scheduling where workers are allowed to perform more than one job simultaneously in different machines, as long as they respect the times they must be present in a previous assigned machine. Notwithstanding, this work was

mainly inspired by manufacturing scheduling problems. In 2001 Felan and Fry [5] investigated the effect of having different levels of training across the workforce. 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. In 2011, Xianzhou and Zhenhe [9], presented a new immune Genetic Algorithm through combining immune algorithm with genetic algorithm which proved to have a higher degree of convergence precision in solving a DRC flexible job shop scheduling problem. Lei and Guo (2014) [6] proposed a VNS composed of two neighbourhood search procedures and a restarting mechanism to minimize the makespan. In 2017 Zhang et al. [11] proposed a Particle Swarm Optimization with a three layered encoding scheme in order to minimize the production period and the production cost.

In 2019 Cunha et al. [4] 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. In 2020 Akbar and Irohara [1] 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 (PGA) and the Modified Bees Algorithm (BA).

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 solely random selection.

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

(3) - An Increment Initialization that combines compatible tests together which improves results by 5%.

(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%.

(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.

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

2. The Dual Resource Constrained Flexible Job Shop Problem

2.1. Application to the Quality Control Laboratory Scheduling

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 operation while the worker is only required at a number of predetermined intervals (tasks) during the processing time of the test. These mandatory presential tasks for the worker are the machine setup (start), the preparation of sample and the materials (medium) and data processing at the end. All of these operations must be performed by the same worker.

2.2. Formulation of the Scheduling Problem

The following mathematical formulation was based on [7]. 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.

2.3. Objective Functions

The Objective function of the problem is to minimise the total completion time J (1) where c_j is the completion time of job j, i.e., the time required to complete the operations of that job.

$$\min \mathcal{J} = \sum_{j \in J} c_j, \quad \forall j \in J \tag{1}$$

2.4. 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:

$$c_j \geq t_{c_j j} + p_{c_j j} \tag{2}$$

Each test can only be assigned to one suitable machine (3) and worker (4):

$$\sum_{k \in K} x_{ijk} = 1, \quad \forall j \in J, i \in O_j, k \in K_{ij} \tag{3}$$

$$\sum_{h \in W} \alpha_{ijk} = 1, \quad \forall j \in J, i \in O_j, h \in W_{ij} \tag{4}$$

Constraint (5) guarantees the precedence is respected between operations of the same job.

$$t_{ij} \geq t_{(i-1)j} + p_{(i-1)j} \tag{5}$$

For the sequencing of operations in machines, constraint (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 ($x_{ijk} = x_{i'j'k}$).

$$t_{ij} \geq t_{i'j'} + p_{i'j'} - (2 - x_{ijk} - x_{i'j'k'} + \beta_{iji'j'})M, \quad \forall j \in J, i \in O_j, O_{ij} \neq O_{i'j'}, k \in K_{ij} \cap K_{i'j'} \tag{6}$$

As the prior constraint is disjunctive, the big M formulation is adopted, Similarly, constraint (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.

$$t_{i'j'} \geq t_{ij} + p_{ij} - (3 - x_{ijk} - x_{i'j'k'} - \beta_{iji'j'})M, \quad \forall j \in J, i \in O_j, O_{ij} \neq O_{i'j'}, k \in K_{ij} \cap K_{i'j'} \tag{7}$$

Two constraints are required to bound for $\beta_{iji'j'} = 0$ and $\beta_{iji'j'} = 1$. In the same way, constraints (8) and (9) ensure that a worker cannot perform overlapping tasks.

$$t_{i'j'} + \rho^s_{ijs} \geq t_{i'j'} + \rho^s_{i'j's'} + \rho^d_{i'j's'} - (2 - \alpha_{ijh} - \alpha_{i'j'h} + \gamma ijsi'j's')M, \quad \forall j \in J, i \in O_j, s \in N_{ij}, O_{ij} \neq O_{i'j'} \tag{8}$$

$$t_{i'j'} + \rho^s_{i'j's'} \geq t_{ij} + \rho^s_{ijs} + \rho^d_{ijs} - (3 - \alpha_{ijh} - \alpha_{i'j'h} - \gamma ijsi'j's')M, \quad \forall j \in J, i \in O_j, s \in N_{ij}, O_{ij} \neq O_{i'j'} \tag{9}$$

3. Proposed Genetic Algorithm

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 have higher chances to propagate their genes to the next generation. In the proposed hybrid GA there are eighth main phases to be considered: (1) The Initial Population, (2) The Encoding and Decoding, (3) The Fitness Evaluation, (4) Selection, (5) Crossover, (6) Mutation, (7) Variable Neighborhood Search and (8) the Dynamic Termination Criteria. Compatibility tables were implemented to assist the correct allocation of the resources to the tasks. This idea was based on [8]. 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.

3.1. Initial Population

Two distinct implementations which rely on different logic were created: the Initial Population Job Randomization (IPJR) and the novel Incremental Initial Population Worker Randomization (IIPWR). In both cases, a limitation in the resources allocation, namely the Prohibition Condition (PC), was

implemented. PC restricts 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.

Initial Population with Job Selection

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.

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 analyst 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 1 was calculated with it's results being the time that the worker has to wait before he can work on the next assigned job.

Table 1: Increment Table

	1.1	2.1
1.1	–	0
2.1	19	–

Table 2: 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
Analyst Selection (AS)	1	1	1	2	1

The number of jobs grouped together in a row is capped at two in a row, since, 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.

3.2. Encoding and Decoding

The implemented encoding scheme has 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, table 2 represents a possible configuration for the encoding mechanism.

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 have a resolution of 0.1 representing the workers' and machine times necessary in each operation. 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 1 for two distinct consecutive operations where the machine is not repeated.

3.3. Fitness Evaluation

The Fitness Evaluation quantifies all individuals in terms of the chosen objective function. A fitness score is assigned to each solution which correlates with the probability that the individual has of being selected for reproduction and thus remaining or not in the gene pool. This process consists of performing the decoding for all the individuals in the population to obtain the time that each job has finished. Afterwards, the fitness value can be calculated de-

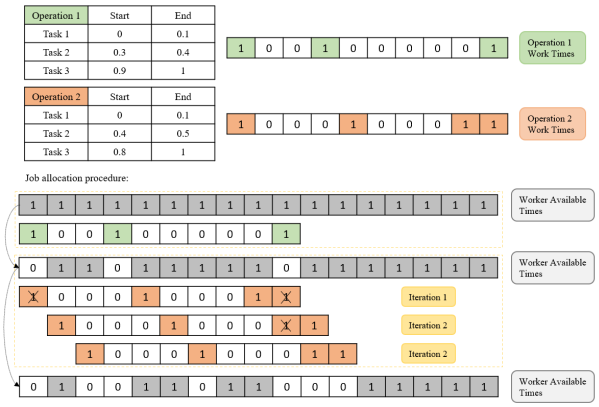

Figure 1: Job allocation example with two consecutive operations

pending on the objective function.

3.4. Selection

The Selection phase is achieved through an Elitism operation (which starts once the number of iterations is superior 1/8 of the total population) and is followed by a Tournament Selection. The Elitism procedure ensures that the 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.

3.5. Crossover

Three different crossover operations are explored: Single Point Crossover, Double Point Crossover and MPX crossover. As previously utilised in the Population Initialization phase, weights were implemented to account for different probabilities of choosing a certain path. After every crossover, feasibility and precedence checking is required.

Single Point Crossover

Choose 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.

Double Point Crossover

Choose two different random numbers (in the chromosome length) which correspond to a certain interval gap in the parents genes. This middle section is swapped between the two to produce both offspring.

3.6. MPX Crossover

Based on [10], the MPX 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.

3.7. Mutation

Offspring population have a certain chance to mutate. The mutation phase starts with a Shift Mutation based on [2] followed by an ”Intelligent Mutation” [6]. Depending on a certain predetermined probability, a machine and worker mutation is performed. Afterwards, the non-mutated off-springs are combined with the mutated ones to form the final off-springs population.

Shift Mutation

The Shift mutation resembles the Double Point Crossover 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.

Intelligent Mutation

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.

Worker and Machine mutation

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

3.8. Variable Neighbourhood Search

VNS explores 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 performed by operations similar to the ones implemented in the mutation phase in section. 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.

3.9. 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 on following iterations it return to the normal values and the counter is returned 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.

4. Results & discussion

4.1. Descripiton of the Experiments

To allow for realistic data, a study by Martins et al. [7] was conducted on real quality control laboratory and an instance generator was developed. This program aims to mimic the conditions faced on a regular working day in this Quality Control Laboratory environment.

Regarding the 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 instances (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.

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.

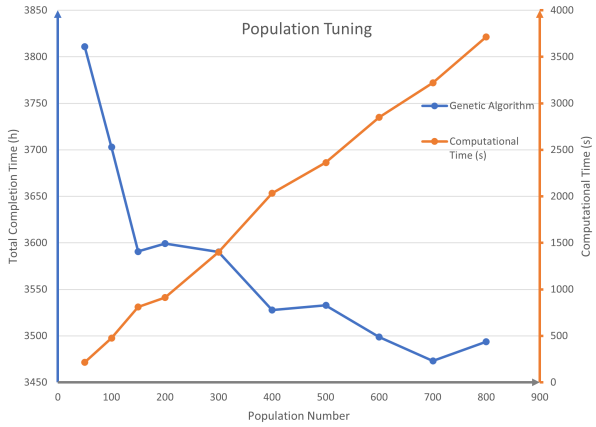

Figure 2: Study of the Total Completion Time with different population sizes

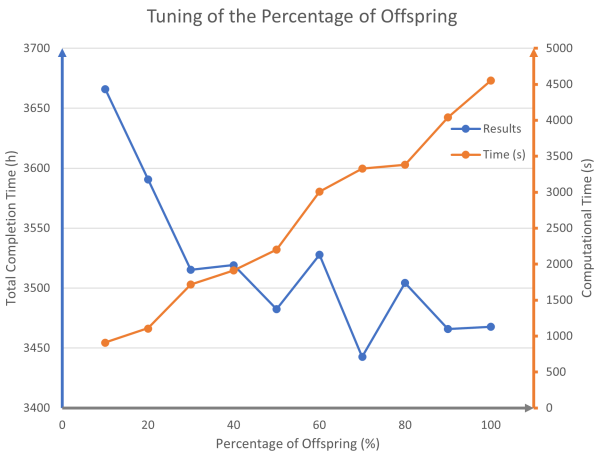

Figure 3: Study of the Total Completion Time with different percentage of Offspring

4.2. General GA Parameters

The general genetic algorithm parameters are considered to be the essential parameters, present in every GA. These following tests ran without VNS and a only one 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 5 best objective functions.

Population Size

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 2. The final results are meant to run the instances for 3h (10800 seconds), therefore, a population 400 off population was selected since it provides a good compromise between the results and the computational time.

Offspring Percentage

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 figure 3, in which the average standard deviation is 62.4 hours. 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, the latter value was considered.

Mutation Percentage

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). Figure 4 shows the results in which the

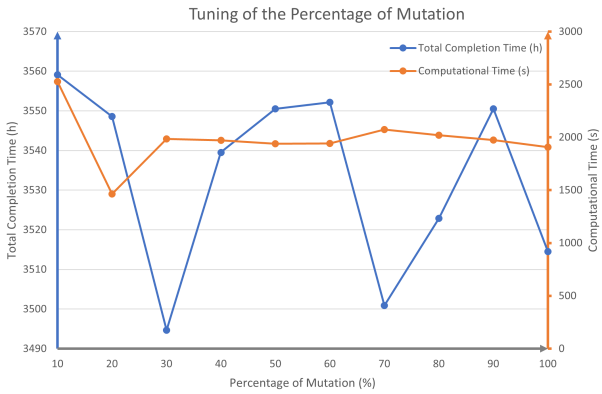

Figure 4: Study of the Total Completion Time with different percentage of mutation

average standard deviation is 67.4 hours. The best percentage of mutation is 30% as it induces the least amount of total completion time in the instance with similar computational time to the remaining results.

4.3. Initial Population

Job Selection Procedure

Tests were performed in all large instances (70 jobs) in order to verify which combination of weights (w_1, w_2, w_3) proved better results. Table 3 presents the sum of all instances for each weight selection. 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. It is worth mentioning that the 70 Job instance is used throughout all remaining tuning tests (unless specified in contrary) as it has the most number of job allocations and consequentially, more reliability, while also being is the target instance of study for this work.

Prohibition Condition Study

The results on table 4 were obtained by performing the average of five runs on a population size of 500

Table 3: Weight Tuning

Weigths	Total Completion Time
100|0|0	**95536**
0|100|0	100871
0|0|100	109048
40|30|30	103146
60|20|20	100805
60|30|10	99976
60|10|30	101799
80|10|10	98488
90|5|0	97031
95|0|5	96009
98|2|0	95759

Table 4: Prohibition Condition Study with different rules

GA	Average of the Total Completion Time
No Prohibition Condition	5398
W = 3, M = 2	**5182**
W = 3, M = 3	5229
W = 3, M = 4	5255
W = 3, M = 6	5269
W = 4, M = 3	5242
W = 4, M = 4	5260
W = 4, M = 5	5275
W = 5, M = 5	5279

for each instance and then averaging all instances for the same coefficients. The average standard deviation across all instances was +/- 259. 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.

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. Table 5 contains a comparison between the sum of the average values of the Normal initialization (IPJS) over five runs with the sum of the average values of this new approach. These are obtained from a population of 500 individuals and only the large instances were considered. The Increment Initialization improves the results by a margin of 5.3% when comparing to the Normal Initialization while also improving the algorithm's convergence speed.

4.4. Crossover Tuning

Tests were performed to the Crossover methods 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 (table 6). The best results were obtained for $c_1 = 10$, $c_2 = 5$, $c_3 = 85$

Table 5: IP with Job Selection (normal initialization) vs Incremental IP with worker selection (increment initialization)

Type of Initial Population	Sum of the Total Completion Time
Normal Initialization	93208.7
Increment Initialization	**88273.7**

Table 6: Crossover Tuning

Crossover Weights	Sum of the Total Completion Time
100|0|0	73931
0|100|0	73189
0|0|100	70439
30|30|40	71849
20|20|60	71497
10|10|80	71222
5|5|90	70071
33.3|33.3|33.4	71711
0|10|90	70935
10|0|90	70093
10|5|85	**69645**
15|5|80	70562
20|5|75	70256

Table 7: 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

4.5. Variable Neighborhood Search

The parameters to tune in the VNS are the following:

(1) - Percentage of best population to undergo VNS.

(2) - 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. The results were plotted and afterwards, the number of occurrences on each percentage was summed and is presented on table 7, indicating that the best percentage of VNS is found between 20 and 30 percent. Therefore, a percentage of 25% for the VNS was chosen. The same procedure was conducted for the number of iterations in the VNS which resulted in a total number of four.

4.6. Final Results Comparison

The final results were compared with a Branch and Cut algorithm from [7] resulting in tables 8, 9 and 10. From table 10 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 simple GA. For small sized instances (8) the Branch and Cut algorithm surpasses both GA's (by 2.3% when compared to the GA + VNS). For medium instances (table 9) the BC presents better results when the number of workers is 2, however, as the instance size increases towards seven workers it looses it's advantage. Therefore, for medium

Table 8: Final Results Comparison for 5 job Instances

Workers	Flex	Rep	GA	GA + VNS	B&C
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

Table 9: Final Results Comparison for 10 job Instances

Workers	Flex	Rep	GA	GA + VNS	B&C
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

sized 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.

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 10 with a total completion time of 41.3 hours (and a makespan of 12.6 hours), it's Gantt Chart for the machine allocation is represented in figure 5.

Table 10: Final Results Comparison for 70 job Instances

Workers	Flex	Rep	GA	GA + VNS	B&C
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	-

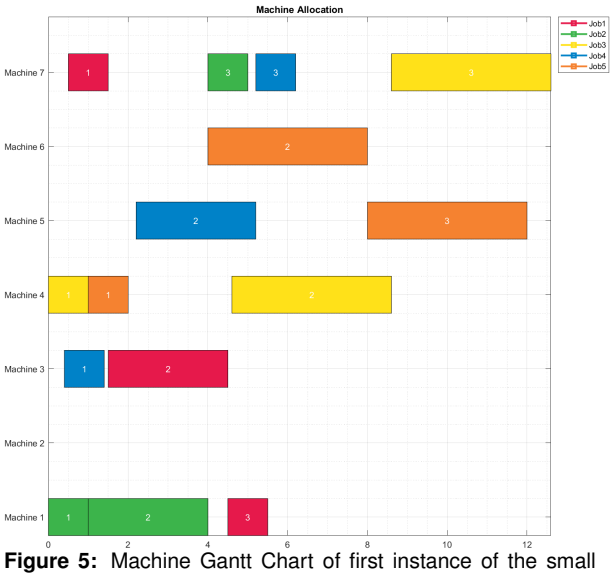

Figure 5: Machine Gantt Chart of first instance of the small instances

5. 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.

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. Also, the prohibition conditions proved fitter results 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 for 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 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 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.

5.1. Future Work

The scheduling of resources extends further than the dual resource constrained flexible job shop problem. Workers do not all perform with the same level of productivity, resources are often not limited to workers and machines, in some cases it may be useful to consider other objective functions and optimization criteria, even simultaneously. Therefore, as of future work, this thesis could be extended to a multi-resource or multi-objective scheduling problem. Also heterogeneous workers may be considered and learning curves implemented in the study.

Also, 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 on this thesis.

On another matter, the perfect tuning of parameters in a genetic algorithm is often something very complicated since these 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 would be of interest to study how the current Genetic Algorithm could be made 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.

References

[1] 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*, 96(March):104004, 2020.

[2] M. Bayão and H. Mesquita. Scheduling of Flexible Job Shop Problem in Dynamic Environment, 2017.

[3] A. Costigliola, F. A. Atde, S. M. Vieira, and J. M. Sousa. Simulation Model of a Quality Control Laboratory in Pharmaceutical Industry. *IFAC-PapersOnLine*, 50(1):9014–9019, 2017.

[4] M. Cunha, L. Viegas, M. Vieira, M. T, and M. Susana. Dual Resource Constrained Scheduling - Quality Control Laboratories. *IFAC PapersOnLine*, 52(13):1421–1426, 2019.

[5] 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*, 39(14):3041–3059, jan 2001.

[6] D. Lei and X. Guo. Variable neighbourhood search for dual-resource constrained flexible job shop scheduling. *International Journal of Production Research*, 52(9):2519–2529, 2014.

[7] M. S E Martins, J. L Viegas, T. Viegas, B. Coito, A. Costigliola, S. M Vieira, J. Sousa, and J. Figueiredo. Minimizing total completion time in large-sized pharmaceutical quality control scheduling, *Submitted and pending approval*.

[8] 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*, 32(3):707–728, 2021.

[9] 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*, 1:42–45, 2011.

[10] M. Yazdani and M. Zandieh. Evolutionary algorithms for multi-objective dual-resource constrained flexible job-shop scheduling problem. *OPSEARCH*, 56(3):983–1006, 2019.

[11] 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*, 28(8):1961–1972, 2017.