Universidad de los Andes

Three solution methods for the uniform parallel machine scheduling problem with time windows, set-up times, unreachable job constraints, and minimizing earliness, tardiness, and total distance

Juan Sebastian Riveros Perez

Supervisor: Ciro Alberto Amaya Guio

Abstract

Three solution methods are proposed for the unaddressed uniform parallel machine scheduling problem with time windows, set-up times, and unreachable job constraints. The concept of unreachable jobs is a rethinking of the Machine Availability Constraints, where a set of available machines is defined for each job. In contrast, Unreachable Job Constraints establish a set of jobs that each machine cannot perform. This new approach reduces the number of sets required as input when the set of jobs is significant. The problem consists of minimizing the number of days a job is performed out of its time window and simultaneously the time taken to travel the total distance required by the solution. An exact method and two heuristic approaches are discussed.

Keywords: Scheduling, Time Windows, Earliness, Tardiness, Distance, Parallel Machines, Heuristic

1. Introduction

This article addresses a new problem inspired by the pineapple crops planting, care and harvesting processes. Pineapple plants need specific applications during their different growth stages; for each application, a time window is specified for when it should be performed. Correctly performed applications provide a good quality fruit.

One of the most common errors in the care process of the pineapple crops is that some of the applications are performed out of the stipulated time windows, and this inconsistency may harm the quality of the final fruit. In this article, the problem is reformulated and presented as a Scheduling problem, and three different methods are introduced as possible solutions.

Here, the main goal is to minimize the number of days a job is performed out of its time window. The proposed methods assume that the effect of an application performed before or after a specific day is equivalent. This assumption can be discarded by adding new parameters to the objective function.

The present article is organized as follows: Section **Error! Reference source not found.** p resents a technical description of the problem and some existing approaches. Section **Error! Reference source not found.** proposes three new methods. Section 4 discusses t he results and a comparison of the methods. Finally, section 5 provides concluding remarks.

2. Problem definition

As stated before, the problem was reformulated as a Scheduling problem. This problem is a new configuration that has not been addressed yet, and according to the $\alpha|\beta|\gamma$ notation introduced by Graham – Lawler – Lenstra - Rinnooy Kan, the problem would be defined as follows:

$$Q_n|s_{kj}, d_j, a_j, U_i|\lambda_1\left(\sum E_j + \sum T_j\right) + \lambda_2\left(\sum s_{kj} \cdot x_{kj}\right)$$

Following the notation, the first field (α) describes the configuration of the machines. Q_n represents uniform parallel machines, i.e., parallel machines with different speeds. In the case study that inspired this article, this configuration is of interest since pineapple crops regularly have two or three different types of machines to achieve the applications; and these machines will provide different speeds to the process. The second field (β) contains the characteristics and constraints of the jobs. s_{kj} models the set-up times; this parameter defines the distance between jobs k and j. a_j and d_j represent the start and end dates of each time window, respectively. U_i is a subset that contains the unreachable jobs for each machine i. Finally, the field (γ) includes the objective function. For this problem, the first part calculates the number of days that a job is performed out of its time window, and the second computes the total distance travelled. This article uses a convex combination of the objective function rather than a Pareto Front because it provides a straightforward solution for given values. However, a Pareto Front can be obtained by varying the importance of the objectives.

Given the particularity of the objective function, first, it is necessary to understand how to implement both earliness and tardiness functions simultaneously. For this, the article by (Garey et al., 1988) proposes symmetric earliness and tardiness penalties. It evaluates the differences between the objective time windows' mid-points and actual processing time windows' midpoints. One of the main conclusions of this article is the advantage of sequencing the jobs with the same objective mid-point M based on their processing time, sequencing the shortest processing time closer to M. To conclude, they used the same idea of the *Shortest Processing Time* rule as optimal for the single machine problem when minimizing the sum of completion times ($1||\sum C_j$).

Like the one proposed by (Hungerländer & Truden, 2018), other articles use the concepts *Forward Dependency* and *Direct Forward Dependency* between time windows. These time windows are contiguous or overlapping with each other. In their paper, (Hungerländer & Truden, 2018) provide a Mixed Integer Linear Programming model, where the primary variable indicates the order in which the customers are served. In the same way, the model by (Bonz, 2021) uses a continuous auxiliary variable to identify the time when a job was performed out of its time window.

Another proposal by (Bonz, 2021) is to build an aggregate objective function as a convex combination of different smaller objective functions. For this approach, each coefficient λ_i represents the weight of its corresponding function. This method evaluates simultaneous objectives and allows a convenient adjustment of their importance.

Finally, (Amaya et al., 2010) developed a model for the *Capacitated Arc Routing Problem with Refill Points and Multiple Loads (CARP-RP-ML)*. They added auxiliary arcs to indicate the end of a cycle on the route, and the refill counts are included as an index of the decision variable. This approach makes it easier to handle extensive paths while creating feasible sub-cycles inside the solution.

All the above approaches served as inspiration for the methods proposed in this article.

3. Solution techniques

An exact method and two heuristics are proposed to solve the above problem. For these methods, it is essential to explain the concept of unreachable jobs U_i. This concept is a rethinking of the Machine Availability Constraints M_j concept, where a set of available machines is defined for each job. However, for the Machine Availability Constraints M_j, more information is required as input for the model; thus, the concept of the unreachable jobs solves this problem since it is assumed that most machines can perform most of the jobs, and unnecessary data can be discarded.

3.1. Exact method – MILP model

The first algorithm is an exact method based on the models developed for the *Vehicle Routing Problem (VRP)*. This technique generates an initial solution and iteratively discards unfeasible solutions with sub-cycles not connected to the warehouse, using a cutting-plane approach.

Sets definition:

I: Set of machines

J_0: Set of jobs

$J = J0 \cup \{j_0\}$: Set of jobs with the warehouse

T: Set of days

U_i: Set of unreachable jobs for the machine i. ($U_i \subset J_0$)

Parameters definition:

p_j: Processing time of job j.

a_j: Minimum expected start day of the job j.

d_j: Maximum expected end day of the job j.

v_i: Speed factor of machine i.

s_{jk}: Time taken to travel from job j to job k.

j_0: Dummy job representing the warehouse.

Q: Hours of a working day.

$p_{j_0} = a_{j_0} = d_{j_0} = 0$

Universidad de los Andes

Primary variable x_{ijkt} and auxiliary variables:

x_{ijkt}: $\begin{cases} 1 & \text{If } j,k \in J \text{ are consecutive in machine } i \in I \text{ on day } t \in T. \\ 0 & \text{d.l.c.} \end{cases}$

y_j: $\begin{cases} 1 & \text{If job } j \text{ is performed before } a_j. \\ 0 & \text{d.l.c.} \end{cases}$

z_j: $\begin{cases} 1 & \text{If job } j \text{ is performed after } d_j. \\ 0 & \text{d.l.c.} \end{cases}$

Day_j: Actual day a job j is done.

$Earl_j$: Number of days a job j was done before a_j.

$Tard_j$: Number of days a job j was done after d_j.

In this model, a job must be done on a working day; thus, it cannot be interrupted once started. On the other hand, dummy variables y_j and z_j linearize the definition of variables $Earl_j$ and $Tard_j$ respectively.

Once the sets, parameters and variables are defined, the formulation is as follows:

Minimize:

$$\lambda_1 \left(\sum_{j \in J_0} Earl_j + Tard_j \right) + \lambda_2 \left(\sum_{t \in T} \sum_{i \in I} \sum_{j \in J} \sum_{k \in J} x_{ijkt} \cdot \frac{s_{jk}}{Q} \right) \quad (1)$$

Subject to:

$$\sum_{t \in T} \sum_{j \in U_i} \sum_{k \in J} x_{ijkt} + x_{ikjt} = 0; \ \ \forall i \in I \quad (2)$$

$$\sum_{t \in T} \sum_{i \in I} \sum_{k \in J} x_{ijkt} = 1; \ \ \forall j \in J_0 \quad (3)$$

$$x_{ijjt} = 0; \ \ \forall t \in T; \ i \in I; \ j \in J$$
$$x_{ijkt} + x_{ikjt} \leq 1; \ \ \forall t \in T; \ i \in I; \ j,k \in J_0 \quad (4)$$

$$\sum_{j \in J} x_{ij_0jt} \leq 1; \ \ \forall t \in T; \ i \in I$$
$$\sum_{j \in J} x_{ij_0jt} = \sum_{j \in J} x_{ijj_0t}; \ \ \forall t \in T; \ i \in I \quad (5)$$
$$\sum_{j \in J_0} \sum_{k \in J_0} x_{ijkt} \leq |J| \cdot \sum_{j \in J} x_{ij_0jt}; \ \ \forall t \in T; \ i \in I$$

$$\sum_{k_1 \in J} x_{ijk_1t} - \sum_{k_2 \in J} x_{ik_2jt} = 0; \ \ \forall t \in T; \ i \in I; \ j \in J \tag{6}$$

$$\sum_{j \in J} \sum_{k \in J} x_{ijkt} \cdot \left(\frac{p_j}{v_i} + s_{jk} \right) \leq Q; \ \ \forall t \in T; \ i \in I \tag{7}$$

$$Day_j = \sum_{t \in T} \sum_{i \in I} \sum_{k \in J} t \cdot x_{ijkt}; \ \ \forall j \in J_0 \tag{8}$$

$$\begin{gathered} a_j - Day_j \leq |T| \cdot y_j; \ \ \forall j \in J_0 \\ Day_j - a_j \leq |T| \cdot (1 - y_j); \ \ \forall j \in J_0 \\ Earl_j \geq a_j - Day_j; \ \ \forall j \in J_0 \\ Earl_j \leq a_j - Day_j + |T| \cdot (1 - y_j); \ \ \forall j \in J_0 \\ Earl_j \leq |T| \cdot y_j; \ \ \forall j \in J_0 \end{gathered} \tag{9}$$

$$\begin{gathered} Day_j - d_j \leq |T| \cdot z_j; \ \ \forall j \in J_0 \\ d_j - Day_j \leq |T| \cdot (1 - z_j); \ \ \forall j \in J_0 \\ Tard_j \geq Day_j - d_j; \ \ \forall j \in J_0 \\ Tard_j \leq Day_j - d_j + |T| \cdot (1 - z_j); \ \ \forall j \in J_0 \\ Tard_j \leq |T| \cdot z_j; \ \ \forall j \in J_0 \end{gathered} \tag{10}$$

$$\begin{gathered} x_{ijkt} \in \{0,1\}; \ \ \forall t \in T; \ i \in I; \ j,k \in J \\ y_j \in \{0,1\}; \ \ \forall j \in J_0 \\ z_j \in \{0,1\}; \ \ \forall j \in J_0 \\ Day_j \geq 0; \ \ \forall j \in J_0 \\ Earl_j \geq 0; \ \ \forall j \in J_0 \\ Tard_j \geq 0; \ \ \forall j \in J_0 \end{gathered} \tag{11}$$

The objective function (1) represents the convex combination of the number of days a job was performed out of its time window and the time taken to travel through the jobs. Constraint (2) ensures that no machine can serve an unreachable. Constraint (3) provides that each job is performed exactly once. The constraints (4) avoid reprocessing the jobs and sub-cycles of two jobs if they are not connected to the warehouse. The set (5) prevents any machine from being used more than once on any day, forces every machine to start from the warehouse and get back to it, and ensures that a used machine must leave first the warehouse that day. Constraint (6) is the flow conservation constraint. Constraint (7) ascertains that the working day hours are not surpassed. Constraint (8) identifies the actual day when a job is done. Sets (9) and (10) are the linearization of constraints (12) and (13), respectively.

$$Earl_j = max\{a_j - Dia_j, 0\} \quad (12)$$

$$Tard_j = max\{Dia_j - d_j, 0\} \quad (13)$$

Lastly, set (11) defines the nature of the variables.

The initial solution obtained with the model above may generate sub-cycles of three or more jobs not connected to the warehouse; therefore, constraint (14) is added iteratively to the model based on the latest solution obtained.

$$\sum_{t\in T}\sum_{i\in I}\sum_{j\in C}\sum_{k\in C} x_{ijkt} \leq |C| - 1 \quad (14)$$

C represents the set of jobs that are not connected to the warehouse. A feasible solution is obtained once set C is an empty set. This *Cutting-Plane* approach discards unnecessary constraints, reducing the model complexity.

3.2. Clustering Heuristic

Since the MILP model above is not a polynomial-time algorithm, only small-size instances can be solved in a reasonable time. For this reason, a heuristical approach was developed.

(Fuentes et al., 2018) proposed a technique to solve the Travelling Salesman Problem. They cluster the cities on k groups using the *k-means* algorithm and use the heuristic *NEH* to obtain a solution. This solution is refined using *Multi-Restart Iterated Local Search (MRSILS)*. This technique served as an inspiration for the proposed method.

The proposed heuristic has two phases: first, it clusters the jobs based on three main features, and second, it applies the exact method proposed above for each cluster.

3.2.1. Phase 1: Clustering

The jobs are partitioned using the *k-means* algorithm based on three features: x coordinate, y coordinate, and the mid-point of the time window associated with each job. This last feature includes both the start and end of the time windows and reduces the amount of information required.

The x and y coordinate features are scaled to avoid asymmetrical weights between these features. The mid-point, however, is not scaled because an intentional skewness is desired to lead the clustering mainly on this feature.

Finally, the number of clusters are computed using the elbow method, varying the values of k, starting in 1, and evaluating the Sum of Squared Errors to their centroids. The error obtained is compared with the one obtained using $k + 1$ clusters, and when the restriction $\frac{e_k - e_{k+1}}{e_{k+1}} < 1$ is met, the number of groups is defined to be $k + 1$.

3.2.2. Phase 2: MILP

Each cluster is assessed if the sum of processing times of the jobs in the cluster is less or equal to the number of hours of a working day times 1.5. This last factor allows the creation of partitions with a coherent number of jobs based on processing times of less than half a working day.

If a cluster does not meet $\sum p_j \leq 1.5 \cdot Q$, the cluster is partitioned following the same approach in phase 1. The exact method is applied once a cluster meets the restriction above.

The constraint (15) is added to the MILP method, where O represents the tuples of days and machines used in other partial solutions. This constraint avoids the superposition of partial solutions, hence constructing an unfeasible final solution.

$$\sum_{j \in J} \sum_{k \in J} x_{ijkt} = 0; \;\; \forall (t,i) \in O \qquad (15)$$

Lastly, all partial solutions are combined to obtain the final solution.

3.3. Dispatch Rule

The last method is based on the dispatch rule *Earliest Due Date (EDD)* and the shortest distance approach, i.e., each job is sequenced if it is the closest job to the previous sequenced job. The solution is obtained as follows:

3.3.1. Define the sets J' and J'^c.

J': Set of sequenced jobs

J'^c: Set of not sequenced jobs

J^p: Set of possible jobs to sequence

3.3.2. Initialization of sets

$J' = \emptyset$

$J'^c = J_0$

$J^p = \emptyset$

3.3.3. Identify the job j^* with the earliest due date d_j in J'^c.

3.3.4. $J^p = \{j \in J'^c : a_j \leq d_{j^*}\}$

3.3.5. Sequence the jobs in J^p using the shortest distance and starting from the warehouse j_0. Pick any of them if two jobs have the same distance from the last sequenced job. The fastest machine will perform these jobs on day d_{j^*}. Once the working day hours are reached for that machine, a new machine must be used. No machine can serve its unreachable jobs U_i.

3.3.6. Update sets J' and J'^c

3.3.7. The jobs that could not be sequenced $j \in J^p \backslash J'$ stay in the not sequenced jobs set J'^c.

3.3.8. Suppose a job that could not be sequenced has the same due date d_j as the last d_{j^*} used, the new $d_{j^*} = d_j + 1$; if not, return to step 3.3.3.

3.3.9. Return to step 3.3.4. until $J'^c = \emptyset$.

4. Results and comparison of the methods

The three methods were applied to different sizes of instances. A group of small, medium and large-size instances were generated. The following points were used when generating the instances:

- A working day has 8 hours for all instances.
- For all instances, λ_1 and λ_2 have the values 0.7 y 0.3, respectively.
- All instances will have 3 or 4 machines with the same probability.
- The set of days is built based on the maximum due date d_j:

$$|T| = \max\left\{\left\lceil max\{d_j\} \cdot 1.1\right\rceil, \left\lceil \sum p_j \right\rceil\right\}$$

- The optimizer used for the MILP model is *Gurobi®*, through *gurobipy* for *Python®*.

Each instance comprises six files:

- Jobs file: This file contains the processing time and the start and end dates of the time window for a given number of jobs. The processing times were generated following an exponential distribution with rate:

$$\lambda = \frac{1}{tAverage}$$

  On the other hand, the time windows are generated randomly with a width of 2 to 10 days, following a discrete uniform distribution.

- Machines file: For a given number of machines, a speed factor is generated following a discrete uniform distribution from 1 to 3. There must be at least one machine with a speed factor of one.

- Unreachable jobs: A Bernoulli experiment with a probability of success of 0.25 is performed for each job in the jobs file. The jobs that get a one (1) are considered unreachable for the machines with a speed factor of one.

- Coordinates file: This file contains the x and y coordinates of each job. These coordinates must be expressed as the time taken to travel x and y distance and keep their direction from the warehouse.

- Distances file: The distance is computed as the rectilinear distance between two points:

$$d_{b1,b2} = |x_{b1} - x_{b2}| + |y_{b1} - y_{b2}|$$

- Parameters file: This file contains the number of days computed above, the number of hours the working day has (Q), and the values for λ_1 and λ_2.

4.1. Small-size instances

Ten small-size instances were generated with values between 16 and 30 jobs:

Instance	Number of jobs	Number of machines
Instance 1	24	4
Instance 2	24	3
Instance 3	20	4
Instance 4	20	4
Instance 5	16	4
Instance 6	16	3
Instance 7	24	4
Instance 8	16	4
Instance 9	24	3
Instance 10	24	4

Table 1. Small-size instances

Each instance was solved using each method, getting the following results:

Instance	OF MILP	OF Heuristic	OF Dispatch Rule	Heuristic Error	Dispatch Rule Error
Instance 1	0.9835	1.0474	1.0163	6.49%	3.33%
Instance 2	0.4338	0.4457	0.4842	2.74%	11.62%
Instance 3	0.8969	0.8969	0.9143	0.00%	1.94%
Instance 4	0.6709	0.6709	0.6796	0.00%	1.30%
Instance 5	1.1917	1.2425	1.3169	4.26%	10.50%
Instance 6	0.6298	0.6298	0.6900	0.00%	9.56%
Instance 7	0.7277	0.7277	0.7447	0.00%	2.34%
Instance 8	0.7492	0.7492	0.8399	0.00%	12.11%
Instance 9	0.9332	1.0150	0.9637	8.76%	3.27%
Instance 10	0.9447	0.9690	1.1579	2.57%	22.56%

Table 2. Small-size instances results

Results are shown in Table 2, where columns 2 to 5 report the objective function values for the MILP, clustering heuristic and dispatch rule methods, respectively. The last two columns provide the errors for each heuristic method, computed as follows:

$$e_H = \frac{OF_H - OF_{MILP}}{OF_{MILP}}$$

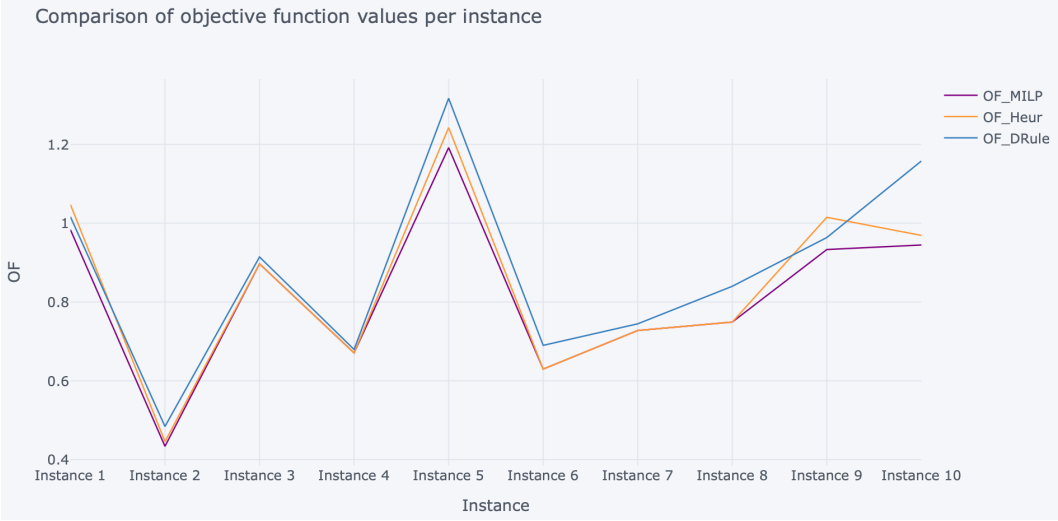

Illustration 1. Comparison of objective function values per instance – Small-size instances

In Table 2 and Illustration 1 the clustering heuristic outperformed the dispatch rule and found the optimal value for five out of the ten instances, while the dispatch rule did not find any. Nevertheless, statistical testing must be performed to come to any conclusion. A histogram is presented below:

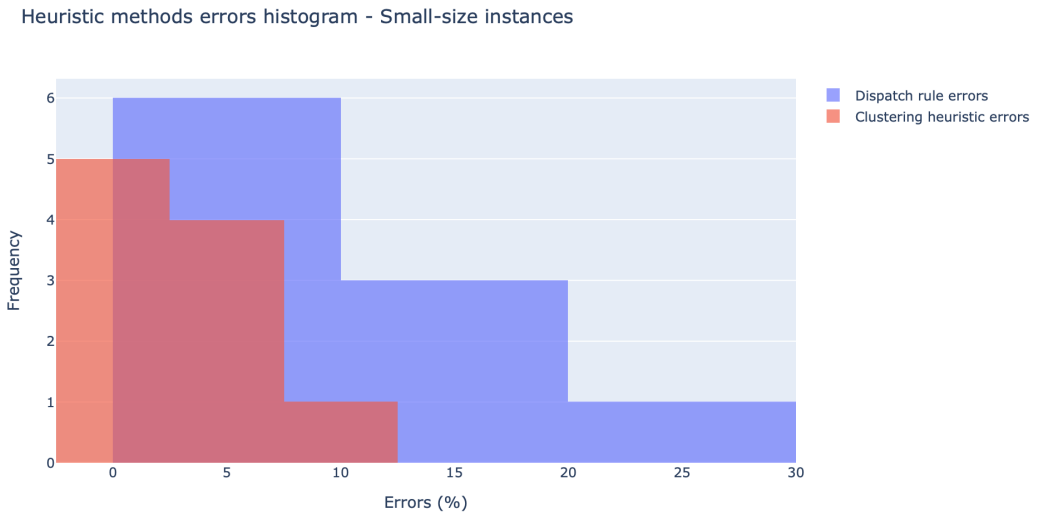

Illustration 2. Heuristic methods errors histogram for small-size instances

The most common test used to compare means is the t-Student test; however, this test requires the data to be normally distributed. Thereby, a Shapiro test is performed; this test has the following hypothesis testing:

H_0: The sample is normally distributed

H_a: The sample is not normally distributed

The p-values 0.019 and 0.06425 were obtained from this test for the clustering heuristic and the dispatch rule errors, respectively. Hence with an alpha level of 0.05, a t-Student test is not viable. Now, given the size of the samples, a normal transformation will not be performed. Instead, a non-parametric test will be conducted, specifically the U Mann-Whitney test, which allows the comparison of any pair of samples regardless of their distributions. For this test, a paired right-tailed test is performed:

$$H_0: \mu_{RD} \leq \mu_H$$

$$H_a: \mu_{RD} > \mu_H$$

Where μ_{RD} and μ_H denote the mean of the errors obtained with the dispatch rule and the clustering heuristic. This test generated a p-value of 0.03223; thus, the null hypothesis is rejected with an alpha level of 0.05, concluding that the clustering heuristic outperformed the dispatch rule for the small-size instances.

It is desired to measure the reduction in time achieved with each heuristic method; this reduction is computed using the time of the exact method as a basis. The times presented below are reported in seconds.

Instance	MILP Time (s)	Heuristic Time (s)	Dispatch Rule Time (s)	Heuristic Reduction	Dispatch Rule Reduction
Instance 1	740	39	1	94.73%	99.86%
Instance 2	4828	8	1	99.83%	99.98%
Instance 3	877	31	1	96.47%	99.89%
Instance 4	111	15	1	86.49%	99.10%
Instance 5	15	3	1	80.00%	93.33%
Instance 6	27	3	1	88.89%	96.30%
Instance 7	5577	11	2	99.80%	99.96%
Instance 8	35	3	1	91.43%	97.14%
Instance 9	17799	25	1	99.86%	99.99%
Instance 10	160	7	1	95.62%	99.38%

Table 3. Small-size instances times

The reduction columns give the percentage of the base time trimmed after using each method. The dispatch rule reduced the time by an average of 98.49%, while the clustering heuristic by 93.31%.

4.2. Medium-size instances

Ten medium-size instances were generated, as well, with values between 48 to 75 jobs:

Instance	Number of jobs	Number of machines
Instance 1	75	3
Instance 2	75	3
Instance 3	60	4
Instance 4	75	3
Instance 5	75	3
Instance 6	48	4
Instance 7	75	3
Instance 8	60	4
Instance 9	60	3
Instance 10	60	4

Table 4. Medium-size instances

Each instance was solved using both heuristic methods. Moreover, since the exact method is not applicable for these instances, a relaxation of the MILP model is proposed to obtain a lower bound to compare. This new Linear Programming model is obtained by replacing the binary variable

$$x_{ijkt} \in \{0,1\};\ \ \forall t \in T;\ i \in I;\ j,k \in J$$

from constraint (11) with the continuous variable in expression (16).

$$x_{ijkt} \geq 0;\ \ \forall t \in T;\ i \in I;\ j,k \in J \qquad (16)$$

This replacement reduces the complexity of the original model and generates a solution in a reasonable time. These objective function values are bound since the model creates unfeasible solutions. Then, a possible optimal value is estimated by calculating the ratios between the bound and the optimal values for the small-size instances.

Instance	OF MILP	OF LP	OF LP/ OF MILP
Instance 1	0.9835	0.0567	5.77%
Instance 2	0.4338	0.0176	4.06%
Instance 3	0.8969	0.0161	1.79%
Instance 4	0.6709	0.0329	4.91%
Instance 5	1.1917	0.0938	7.87%
Instance 6	0.6298	0.0420	6.67%
Instance 7	0.7277	0.0107	1.46%
Instance 8	0.7492	0.0525	7.01%

Instance 9	0.9332	0.0413	4.42%
Instance 10	0.9447	0.0291	3.08%

Table 5. LP and MILP ratios for small-size instances

After performing a normality test, a p-value of 0.7992 is obtained; consequently, the ratio between the LP and MILP can be said to be normally distributed with a mean of 4.70% and a standard deviation of 2.17%. Then the optimal value is estimated as follows:

$$FO_{MILP} \approx \frac{FO_{LP}}{E(FO_{LP}/FO_{MILP})} = \frac{FO_{LP}}{\mu_{LP/MILP}}$$

This estimate does not provide the accurate optimal value but reduces the magnitude of the errors obtained after using the LP objective values when calculating the errors.

Results below:

Instance	OF LP	OF MILP	OF Heuristic	OF Dispatch Rule	Heuristic Error	Dispatch Rule Error
Instance 1	0.0526	1.1195	2.8187	2.8054	151.77%	150.59%
Instance 2	0.0069	0.1467	2.0313	1.6689	1,284.24%	1,037.27%
Instance 3	0.0272	0.5779	1.2009	1.1475	107.81%	98.57%
Instance 4	0.0278	0.5918	1.9617	1.5148	231.50%	155.98%
Instance 5	0.0455	0.9671	1.7879	1.6161	84.87%	67.10%
Instance 6	0.0283	0.6014	1.2581	1.1520	109.21%	91.56%
Instance 7	0.0646	1.3746	2.2505	1.7784	63.72%	29.37%
Instance 8	0.0270	0.5747	2.0379	1.8398	254.59%	220.12%
Instance 9	0.0335	0.7130	2.5334	2.1680	255.30%	204.05%
Instance 10	0.0737	1.5677	2.7612	2.0374	76.13%	29.96%

Table 6. Medium-size instances results

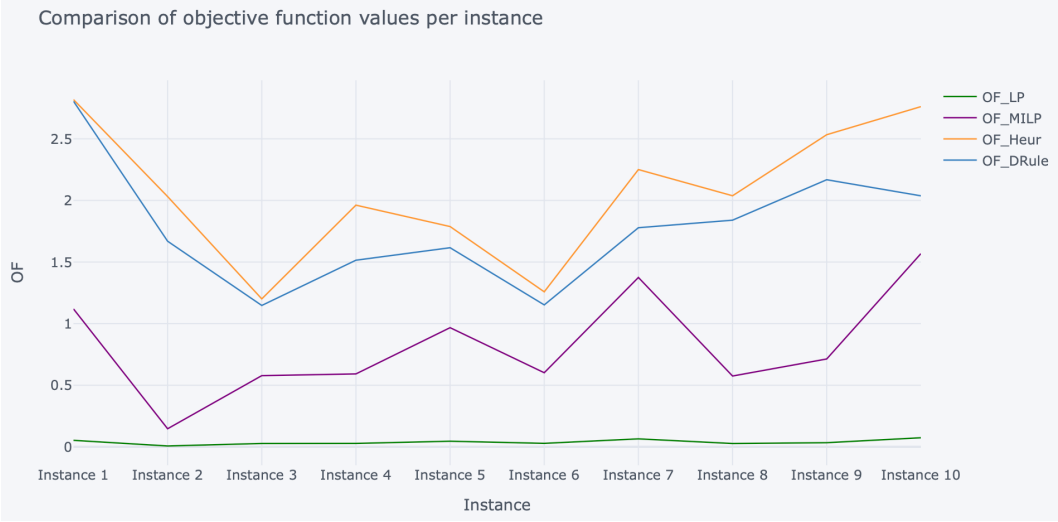

Illustration 3. Comparison of objective function values per instance – Medium-size instances

Then it seems that the dispatch rule outperformed the clustering heuristic for the medium-size instances. The following left-tailed hypothesis testing can corroborate this intuition:

$$H_0: \mu_{RD} \geq \mu_H$$

$$H_a: \mu_{RD} < \mu_H$$

A p-value of 0.0009766 indicates that the dispatch rule indeed outperformed the clustering heuristic for this set of instances.

4.3. Large-size instances

Five large-size instances were generated with values between 140 and 210 jobs:

Instance	Number of jobs	Number of machines
Instance 1	210	3
Instance 2	175	3
Instance 3	175	4
Instance 4	140	3
Instance 5	140	4

Table 7. Large-size instances

For these instances, neither the MILP nor the LP models were plausible because the number of combinations was too high. The results obtained were:

Instance	OF Heuristic	OF Dispatch Rule	Heuristic Time (s)	Dispatch Rule Time (s)
Instance 1	6.9615	5.6110	55	6
Instance 2	8.5625	7.0461	130	5
Instance 3	5.3463	6.1881	115	5

Instance 4	3.7899	3.1141	259	4
Instance 5	5.3953	3.8180	1,843	3

Table 8. Large-size instances results

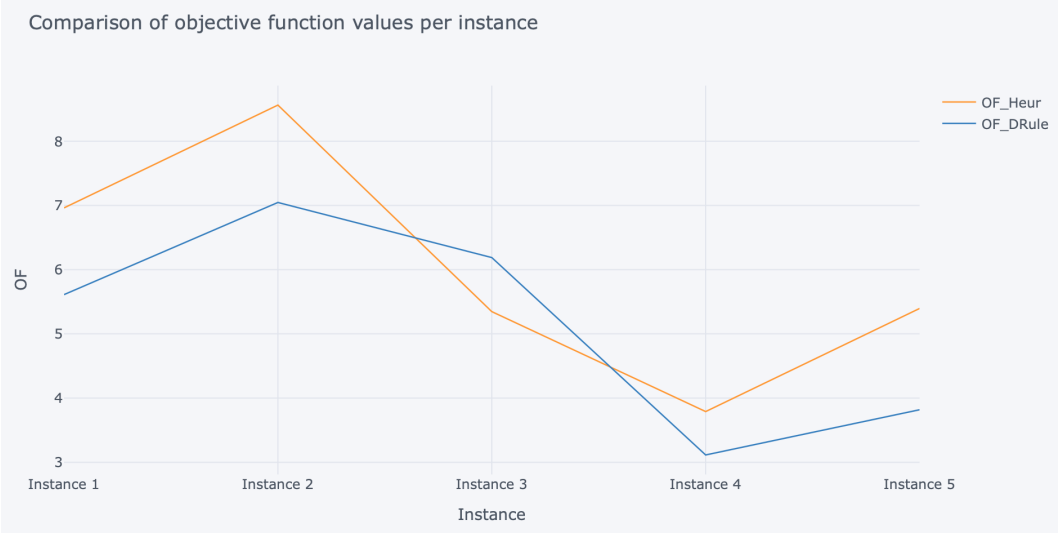

Illustration 4. Comparison of objective function values per instance – Large-size instances

Therefore, the dispatch rule outperformed the clustering heuristic in almost every instance.

The clustering heuristic had an excellent performance on the small-size instances, but the dispatch rule produced better solutions for bigger ones. This behaviour occurs due to the error introduced by partitioning the jobs in the clustering heuristic, considering that with more clusters, the odds of cutting an optimal route are higher.

The graphic below depicts the impact of the number of clusters in the objective function value and the running time for small-size instances with the clustering heuristic:

Illustration 5. Objective function vs number of clusters – Clustering heuristic

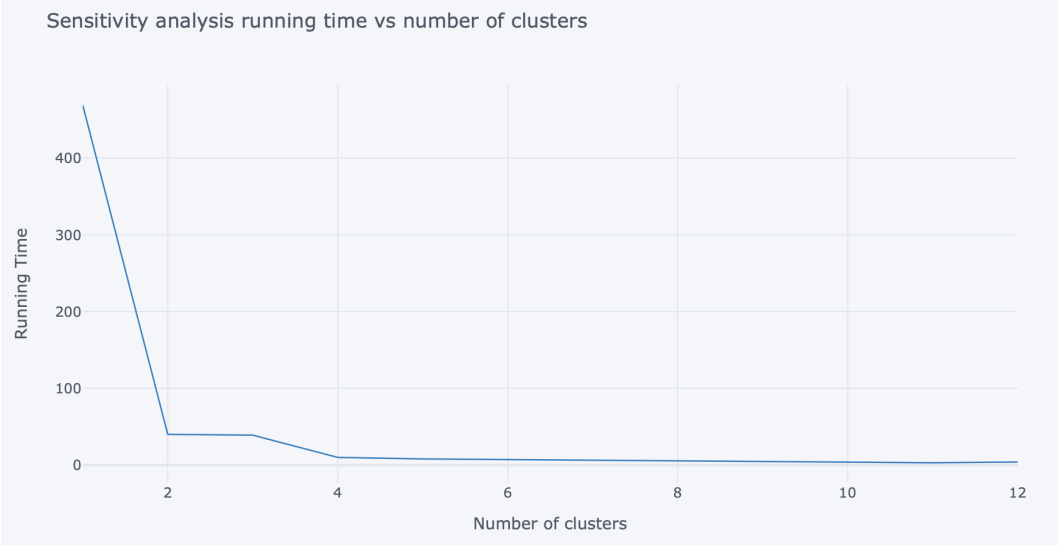

Illustration 6. Running time vs number of clusters – Clustering heuristic

A p-value of 0.1356 proves that the variation of the objective function after adding one cluster to the heuristic follows a normal distribution with a mean of 0.0490 and a standard deviation of 0.0742. Thus, for each partition added to the set of jobs, the objective function value increases by an average of 0.0490. In this way, larger-size instances may require more clusters to run in a reasonable time so that the error will be more significant for these instances.

5. Conclusions

Based on the results above, the clustering heuristic reduces the running times effectively. However, given the number of clusters required to get a group small enough to apply the MILP model, the error introduced increases rapidly for each partition added. Thus, fewer clusters indicate better results. That is why the clustering heuristic performed the best on the small-sized instances and not on the bigger ones.

When comparing the objective function values per instance: Illustration 1, Illustration 3 and Illustration 4, it is observable that the gaps between the orange and blue lines, clustering heuristic and dispatch rule values are broader for the medium and large-size instances.

At the same time, as outlined in Illustration 7 the number of clusters is not the only variable that defines the algorithm's running time because two instances of the same size can produce very different times. This behaviour might be related to the closeness between jobs, i.e., when a set of jobs are too tight together, the *k-means* algorithm will place them in the same cluster, leaving some clusters with more jobs than others, producing longer running times when running the MILP model on these clusters, than instances with smaller clusters, even with more clusters.

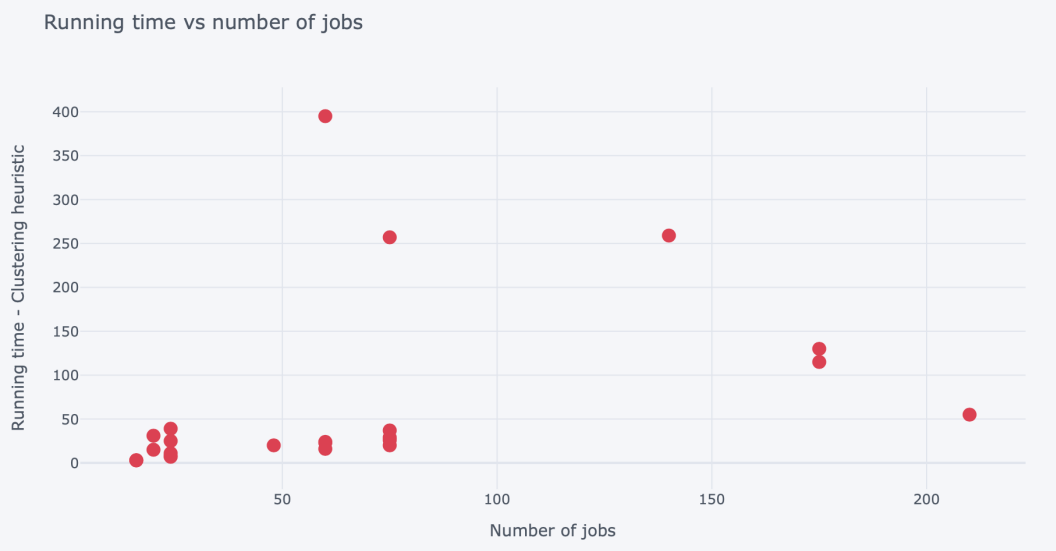

Illustration 7. Running time vs number of jobs – Clustering heuristic

Finally, as represented in Illustration 8, for the dispatch rule, the running time and the number of jobs have a stronger tendency, which appears to be linear. Besides, given that the complete set of jobs is used to generate the solution with the dispatch rule, the error introduced do not increase based on features other than the size of the instance.

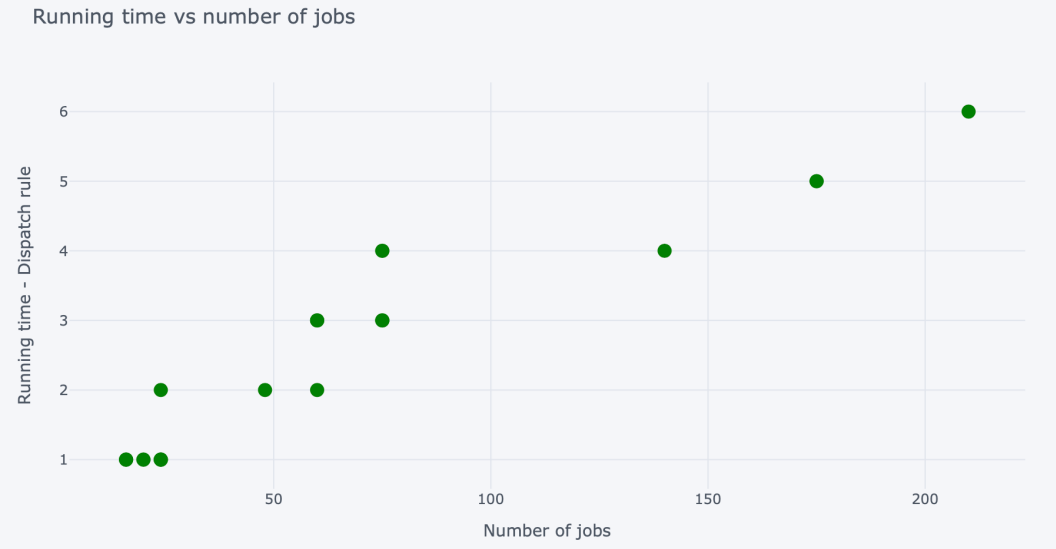

Illustration 8. Running time vs number of jobs – Dispatch rule

Thus, it might be implied that the dispatch rule may provide a better solution for larger-size instances.

It is proposed to readjust the model used for the second phase of the clustering heuristic in future work. If the time of the second phase can be reduced, the errors introduced by the number of partitions will decrease since fewer clusters will be required to get the same time.

6. References

Amaya, C. A., Langevin, A. A., & Trépanier, M. M. (2010). A heuristic method for the capacitated arc routing problem with refill points and multiple loads. *Journal of the Operational Research Society*, *61*(7), 1095–1103. https://doi.org/10.1057/jors.2009.58

Bonz, J. (2021). Application of a multi-objective multi traveling salesperson problem with time windows. *Public Transport*, *13*(1). https://doi.org/10.1007/s12469-020-00258-6

Fuentes, G. E. A., Gress, E. S. H., Mora, J. C. S. T., & Marín, J. M. (2018). Solution to travelling salesman problem by clusters and a modified multi-restart iterated local search metaheuristic. *PLoS ONE*, *13*(8). https://doi.org/10.1371/journal.pone.0201868

Garey, M. R., Tarjan, R. E., & Wilfong, G. T. (1988). ONE-PROCESSOR SCHEDULING WITH SYMMETRIC EARLINESS AND TARDINESS PENALTIES. *Mathematics of Operations Research*, *13*(2). https://doi.org/10.1287/moor.13.2.330

Hungerländer, P., & Truden, C. (2018). Efficient and Easy-to-Implement Mixed-Integer Linear Programs for the Traveling Salesperson Problem with Time Windows. *Transportation Research Procedia*, *30*. https://doi.org/10.1016/j.trpro.2018.09.018