LABORATOIRE DES SCIENCES DU NUMÉRIQUE DE NANTES

Nantes Université

Optimising mail sorting: a case study of the French company La Poste

Supervised Research Project (TER)

Nicolas COMPÈRE – Adrien PICHON

May 2023

Nantes University — UFR Sciences et Techniques
Computer Science Master in Optimisation in Operational Research (ORO)
Academic Year 2022-2023

Contents

1 Introduction

This work has been carried out as a supervised research project in the context of our MSc in Optimization and Operations Research. The duration of the project was 5 months, from January to May. We were a group of two students, Nicolas Compère[1] and Adrien Pichon[2] and we were supervised by professor Evgeny Gurevsky[3] from Nantes University. We had weekly meetings with our supervising professor to check our progress and discuss our ideas for improvements.

1.1 Context

1.1.1 La Poste

La Poste has historically been the mail routing and distributing operator in France since 1991. It delivers nearly 3 billion packages and 9 billion mails throughout the world each year, making it the number one operator in Europe. Throughout the years, the volume of mail exchanges around the world has diminished drastically, compelling historic mail operators to adapt. La Poste, the French public mail distribution operator, has seen its mail volume decline from 18 billion in 2008 to 9 billion in 2018, and its projections point to 5 billion in 2025.

As the old system was optimized for a fixed number of mail throughout the week, the group La Poste has to develop new mail sorting processes in order to adapt to the declining mail traffic.

1.1.2 Mail sorting

In the current sorting method, the mail batch that is assigned to each postman is sorted in the order of the geographical position of the mailboxes. This is meant to facilitate the postmen's delivery rounds. La Poste has put in place some new processes in its Industrial Mail Platforms (PIC) to replace the manual mail sorting that was done by the postmen in the post office. Industrial mail sorting machines carry out the mail sorting nowadays. Several sorting steps are required because of the limited number of outputs of the machines (multi-passage sorting). The sorted mail is assigned to a batch corresponding to a distribution round. At the end of all sorting steps, each batch corresponds to one postman's round. There can sometimes be more than one batch depending on the volume of mail to deliver.

Our research only focuses on a small portion of the 2-phase sorting but we are going to briefly explain how it works. The goal is to assign each mail to a batch based on the geographical location of the mail box. This is illustrated in the following figures [1].

[1]nicolas.compere@etu.univ-nantes.fr
[2]adrien.pichon1@etu.univ-nantes.fr
[3]evgeny.gurevsky@univ-nantes.fr

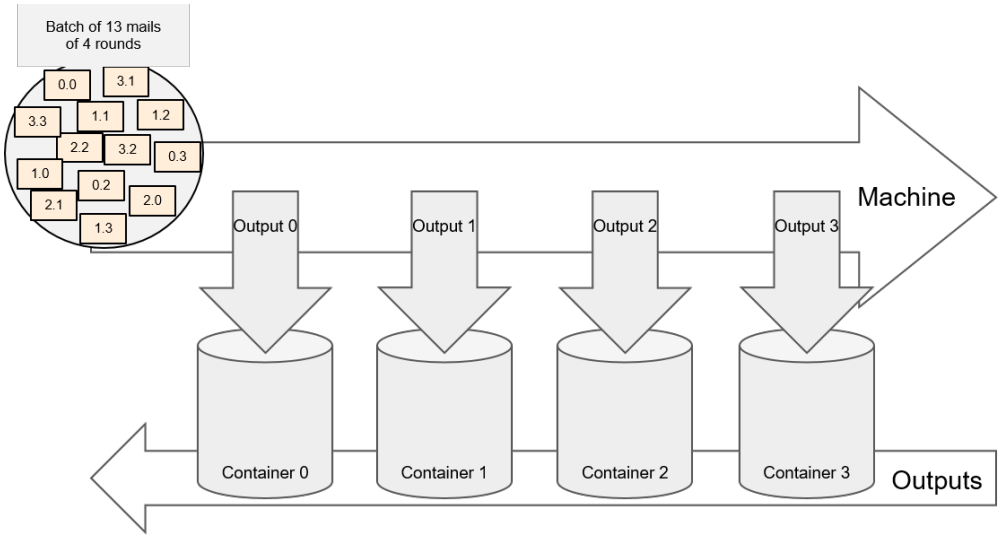

Figure 1: Initialization of Phase 1

In this example, we have 13 mails of 4 rounds. The sorting machine has 4 outputs from 0 to 3. A container is assigned to each output, corresponding to the rank of the position of the mail box.

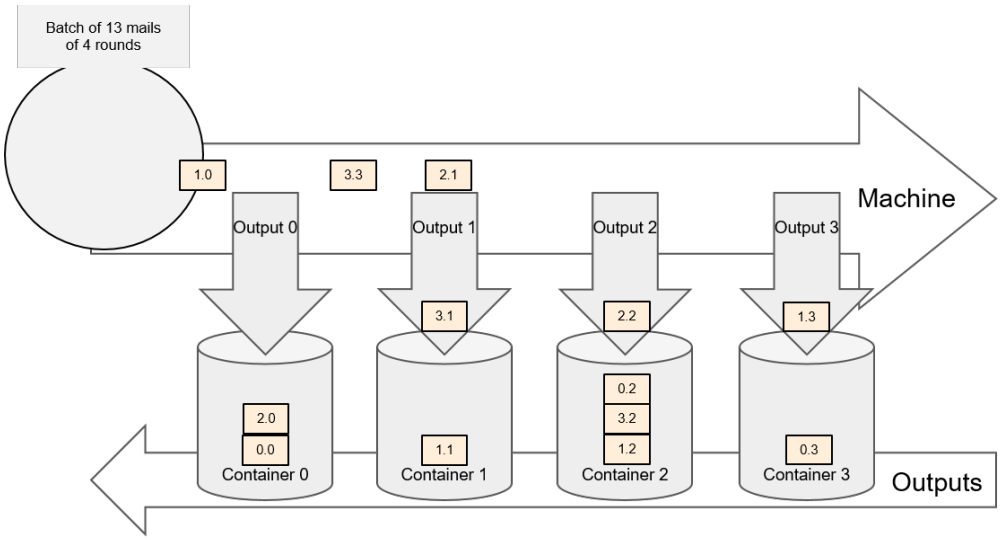

Figure 2: Intermediate step of Phase 1

We assign each mail to a container and we end up obtaining an output like this :

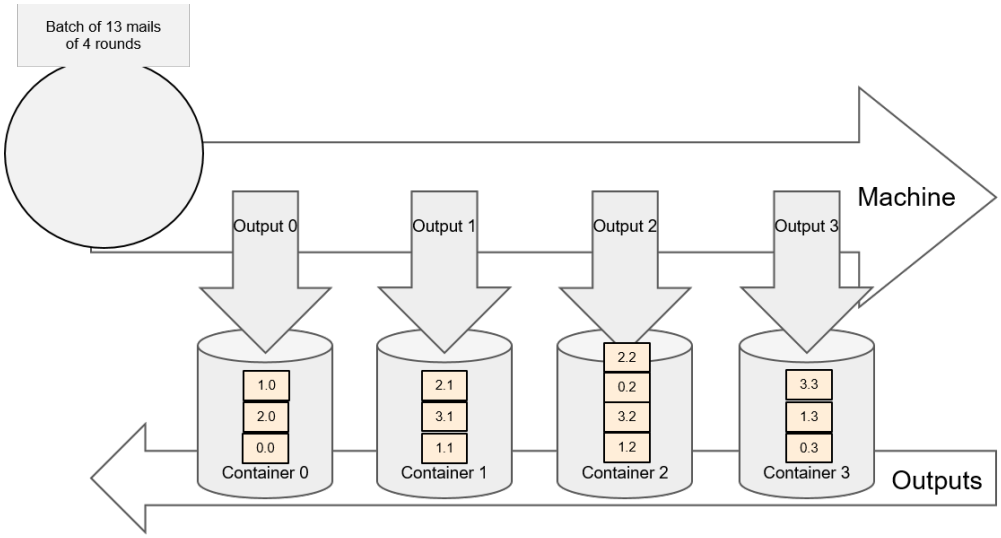

Figure 3: End of Phase 1

Once phase 1 is complete, we place the containers back into the input of the machines to realise the second step.

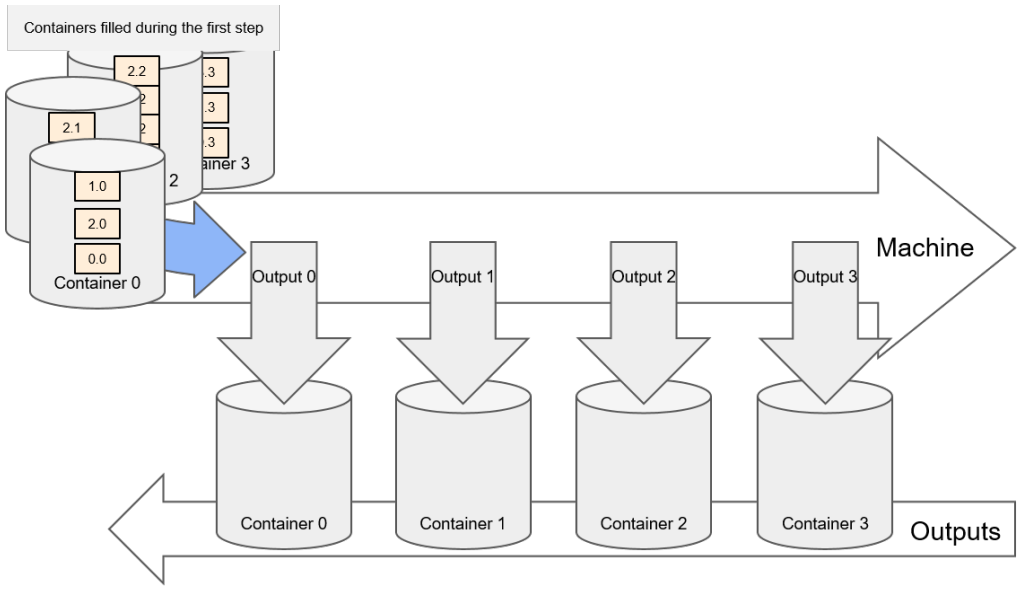

Figure 4: Initialization of Phase 2

Thanks to Phase 1, we are certain that every mail is arriving in the correct order for Phase 2. Mail that is meant to be in first position arrives first in the machine with Container 0 then all mail in position 2 arrives with Container 1 and so on.

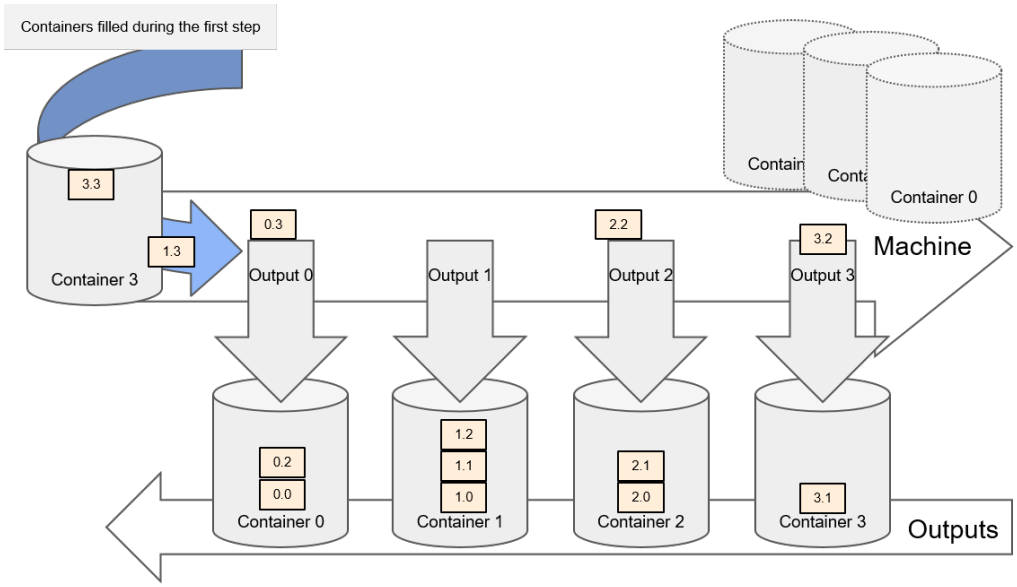

Figure 5: Intermediate step of Phase 2

At the end of Phase 2, we get all mail sorted for each round.

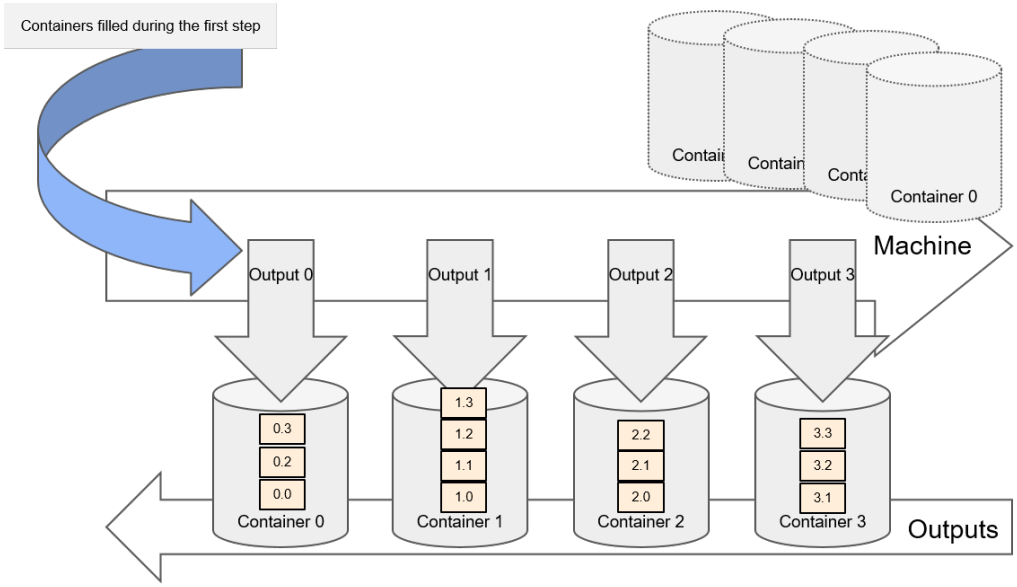

Figure 6: End of phase 2

The problem with this method is that we sometimes get very unbalanced outputs, with some being very loaded and others being empty. This heavily affects the performances of the machines.

1.2 Goal of our research

Our research is focusing on what happens between Phase 1 and 2 of the 2-step sorting. The objective is to find an efficient algorithm that is able to even out mail batch volumes in short time. This would allow an easier handling of the mail batches for the workers at the industrial mail platform, as loaded batches can be heavy and thus harder to handle.

To do this, we first wanted to check how a mixed-integer program would perform. Because we knew from the start that a solver using a MILP wouldn't perform well in CPU time, especially with larger data files, we very quickly began to explore a simulated annealing like heuristic algorithm. We started with a first version given by our supervising professor. The main goal of our research was to improve the existing algorithm and also to explore different approaches of the heuristic.

Figure 7: Sorting machines of the postal industrial platform (PIC) near Rennes

2 Literature review

There isn't much work that has been done specifically about mail batch balancing but we can compare it to the Assembly Line Balancing Problem and our final algorithms take ideas from the Simulated Annealing Heuristic.

2.1 Assembly Line Balancing Problem

This problem can be considered as a variant of the well-known assembly line balancing problem (ALBP). It is a well known problem, very popular in the industrial domain. The first mathematical formulation of the problem was written by M. E. Salveson in 1955 [2]. Since then, a lot of industrial production problems were based on it in order to find an optimal solution for recurrent load balancing problems on production lines in the industry.

The problem assigns a number of tasks $I = 1, ..., i, ...|I|$ to m parallel machines $M = 1, ..., k..., m$. Precedence relations between tasks can be illustrated by a precedence graph, with an arc (i, j) if task j cannot be executed before the end of task i. All tasks assigned to each machine k, I_k, are executed in turn and $T_k = \sum_{i \in I_k} t_i$ with t_i the execution time of task i. The goal is to assign each task to a machine by respecting precedence constraints and optimizing a predefined balancing criteria. We can naturally apply this formulation to our problem, which is balancing the volume of our mail batches before Phase 2.

2.2 Simulated Annealing Heuristic

Simulated Annealing is an optimization algorithm. It's a metaheuristic, meaning that it draws inspiration from a real world process. Here, it is based on the physical process of annealing in metallurgy, where a material is heated and gradually cooled to reduce defects and achieve a more stable state. As mentioned by P. J. M. van Laarhoven and E. H. L. Aarts [3], it was first proposed in 1983 by Kirkpatrick et al. [4]. The work was based on the principal introduced by Metropolis in 1953.

The idea is to accept non improving solutions with a certain probability and pursue the research until facing a stop condition. Let S the current solution and S' a neighbor solution to S. We have different rules:

- S' improving S or S^* will always be accepted
- S' too bad regarding S will never be accepted
- S' not improving S might still be accepted

The probability of acceptation is a function, called Boltzmann distribution, expressed by $e^{-\Delta/T}$. This function decreases with $\Delta = z(S') - z(S)$, the magnitude of degradation and T the time already spent in research. The temperature T is controlled by a cooling scheme $\mathcal{T}$.

Our algorithms takes the idea of accepting some degrading solutions in order to get away from local minimal solutions but without the probability of accepting or refusing degrading new solutions.

3 Mixed Integer Linear Programming

We first wanted to try a mixed-integer linear program (MILP). Even though we knew it would not be efficient, especially on large inputs, we wanted to be able to compare the performances with the heuristic that we implemented next.

The MILP model was given to us by our supervising professor and comes from the unpublished Ifac 2023 paper [5].

3.1 Formulation

The model aims at minimizing the linear objective of $L_{\max} - L_{\min}$ which consist of the difference between the most and least loaded outputs of the machine. Later in the paper we refer this objective as f_1. The complete formulation is as follows:

Notations:

- $R = \{1, \ldots, |R|)\}$ is the set of all postmen's rounds. $|R|$ is also served as the number of outputs, used in the second mail sorting step;
- $B^{(r)} = \{1, \ldots, |B^{(r)}|)\}$ is the set of mail batches for the round $r \in R$;
- $v_j^{(r)}$ is the volume of the j-th mail batch in the round $r \in R$. Here, $j \in B^{(r)}$.
- $O = \{1, 2, \ldots, |O|\}$ is the set of outputs for the first mail sorting step;
- $O_j^{(r)} = \left\{j, j+1, \ldots, |O| - |B^{(r)}| + j\right\}$ is the interval of potential outputs for the j-th mail batch in the round $r \in R$;
- $U_k^{(r)} = \{j \in B^{(r)} : k \in O_j^{(r)}\}$ is the set of mail bathes of round r, which can be potentially assigned to the output $k \in O$.

Variables:

- $x_{jk}^{(r)}$ is equal to 1 if the j-th mail batch of round r is assigned to the output $k \in O_j^{(r)}$, 0 otherwise;
- $L_{\min} \geq 0$ is the minimal load per output;
- $L_{\max} \geq 0$ is the maximal load per output.

Model:

$$\min \quad f_1 := L_{\max} - L_{\min} \tag{1}$$

subject to:

$$\sum_{k \in O_j^{(r)}} x_{jk}^{(r)} = 1, \quad \forall j \in B^{(r)}, \quad \forall r \in R \tag{2}$$

$$\sum_{j \in U_k^{(r)}} x_{jk}^{(r)} \leq 1, \quad \forall k \in O, \quad \forall r \in R \tag{3}$$

$$\sum_{k \in O_j^{(r)}} k \cdot x_{jk}^{(r)} \leq \sum_{k \in O_{j+1}^{(r)}} k \cdot x_{j+1,k}^{(r)}, \quad \forall j \in B^{(r)} \setminus |B^{(r)}|, \quad \forall r \in R \tag{4}$$

$$L_{\min} \leq \sum_{r \in R} \sum_{j \in U_k^{(r)}} v_j^{(r)} \cdot x_{jk}^{(r)} \leq L_{\max}, \quad \forall k \in O \tag{5}$$

$$x_{jk}^{(r)} \in \{0,1\}, \quad \forall j \in B^{(r)}, \quad \forall k \in O_j^{(r)}, \quad \forall r \in R$$

$$L_{\min}, \; L_{\max} \geq 0$$

The objective function (1) is minimizing the difference between the most loaded and the least loaded mail batch output. The constraints are as follows:

(2) Ensures that each mail batch j of round r is assigned to exactly one output

(3) For a given round, there is at most one mail batch per output

(4) Precedence mail constraint for each round r

(5) The total load per output has to be between $L_{\min}$ and $L_{\max}$, the least and most loaded outputs

3.2 Other fitness functions

We have come up with other fitness functions that allow us to measure if the mail batch outputs are smoothed out or not.

$$f_2 = \frac{1}{|O|} \sum_{k=1}^{|O|} |C^* - C_k|$$

$$f_3 = \sqrt{\frac{\sum_{k=1}^{|O|} (C^* - C_k)^2}{|O|}}$$

with C^* being the mean of C_k, which is the sum of the k-th batch:

$$C^* = \frac{1}{|O|} \sum_{k=1}^{|O|} C_k$$

f_2 is the quadratic variance and f_3 is the standard deviation of the output volumes. These functions allow us to have another measure beside f_1 and will be used in the following heuristic algorithms as objective functions.

4 Opti-Move heuristic

In this section, we are going to see the original version of the heuristic that we worked on. This version was handed to us with the objective of observing it's performance and look for optimisation and improvements. It also comes from the unpublished Ifac 2023 paper [5].

4.1 Description and Visualisation

Let's take Table 1 that shows an example of four rounds with five outputs.

Round	Out. 1	Out. 2	Out. 3	Out 4	Out 5
1	10	7	3	8	4
2	5	6	1	7	
3	8	9	3		
4	11	4	3		
$\sum$	34	26	10	15	4

Table 1: An example of allocation of mail batches to outputs with $f_1 = 30$

To balance the total loads of the outputs, for each round, we need to move the load of a mail batch to an empty cell, i.e. an empty output in the same row. For example, for round 2, the fourth mail batch can be moved to output 5, the only one in the same row, which is empty. However, the precedence constraints between mail batches must be respected. For round 3, a feasible configuration could be: first mail batch in output 1, second mail batch in output 3 and third mail batch in output 5 leading to the configuration in Table 2. For round 3, a configuration where the first mail batch is in output 1, the second mail batch in output 5 and the third mail batch in output 3 is not feasible because the second mail batch must always precede the third. Nevertheless, as the precedence constraints are circular, another feasible configuration for round 3 could be: third mail batch in output 1, first mail batch in output 2 and second mail batch in output 3.

Round	Out. 1	Out. 2	Out. 3	Out 4	Out 5
1	10	7	3	8	4
2	5	6	1		7
3	8		9		3
4	11	4	3		
$\sum$	34	17	16	8	14

Table 2: Another allocation of mail batches to outputs: $f_1 = 26$

4.2 Opti-Move formulation

Notations:

- $O^{(r)}(s)$ is the set of non-empty outputs in round r of solution s, where each of which has at least one direct right or left empty output. For the example presented in Table 2, $O^{(2)}(s) = \{3, 5\}$.

- $E^{(r,p)}(s)$ is the set of direct left and right empty outputs of the non-empty output p in round r of solution s. For the example presented in Table 2, $E^{(3,3)}(s) = \{2, 4\}$, $E^{(4,3)}(s) = \{4, 5\}$, $E^{(3,1)}(s) = \{2\}$ and $E^{(2,2)}(s) = \emptyset$.

Algorithm 1 Procedure NEIGHBOR

Input: An initial solution $s^{(0)}$ and a tolerance value τ.
Output: A potentially new solution better than $s^{(0)}$.

1. Set $i := 0$ and $r := 1$.
2. Compute $O^{(r)}(s^{(i)})$ and set $P := O^{(r)}(s^{(i)})$.
3. If $P = \emptyset$, then go to Step 5. Otherwise, choose the most loaded output p from P and the least loaded output q from $E^{(r,p)}(s)$.
4. Let $s^{(T)}$ be a solution obtained from solution $s^{(i)}$ by shifting the mail batch from output p to output q for round r. If $f(s^{(T)}) < f(s^{(i)}) + \tau$, then move to a new current solution, *i.e.*, set $i := i + 1$, $s^{(i)} := s^{(T)}$.
5. If $r < |R|$, then set $r := r + 1$ and go to Step 2. Otherwise, stop and return solution $s^{(i)}$.

Algorithm 2 Heuristic OPTI-MOVE

Input: A current solution s.
Output: The best known solution $s^{(B)}$.

- **Stage 1**.

  1.1. Set tolerance value $\tau := 0$ and $s^{(B)} := s$.

  1.2. Let s^* be a solution provided by the procedure NEIGHBOR(s, τ). If $f(s^*) < f(s^{(B)})$, then, set $s^{(B)} := s^*$ and repeat Step 1.2.

- **Stage 2**.

  2.1. Set tolerance value $\tau := 0.05 \cdot f(s)$ and decrement step $\Delta := 0.02$.

  2.2. Let s^* be a solution provided by the procedure NEIGHBOR(s, τ). If $f(s^*) < f(s^{(B)})$, then set $s^{(B)} := s^*$ and repeat Step 2.2.

  2.3. Decrease the tolerance value $\tau := \tau - \Delta$ and reset $s := s^*$. If $\tau > 0$, then go to Step 2.2.

- **Stage 3**.

  3.1. Set tolerance value $\tau := 0$.

  3.2. Let s^* be a solution provided by the procedure NEIGHBOR(s, τ). If $f(s^*) < f(s^{(B)})$, then set $s^{(B)} := s^*$ and repeat Step 3.2. Otherwise, stop and return the best found solution $s^{(B)}$.

4.3 Opti-Move analysis

Results can be found in section 8.3.

Even though execution times are good, a lot better than what we have with the MILP, the results are still not acceptable. We can also observe that it is hard to predict which instances are going to perform well when we look at the results for the OPTICLASS instances in Table 9.

Following these results, we looked into what could be improved in this version of the algorithm. We noticed that during execution, Algorithm 3 only has one degrading phase and only begins to look for improving solutions at the end of said phase, which greatly limits the exploration of the solution space. (This is the normal behavior of such heuristic but it may limit too much in our case).

5 Improved Opti-Move heuristic

This new version of the Opti-move heuristic received an updated strategy to better explore good solutions during the degrading phase (phase 2)

5.1 Improved Opti-Move formulation

The NEIGHBOR Procedure is still the same as only the Opti-Move heuristic received changes in this part. See Section 6 for an update on the neighborhood method.

Notations:

Algorithm 3 Improved OPTI-MOVE

Input: A current solution s and a number of steps N
Output: The best known solution $s^{(B)}$.

- **Stage 1**.

  1.1. Set tolerance value $\tau := 0$ and $s^{(B)} := s$.

  1.2. Let s^* be a solution provided by the procedure NEIGHBOR(s, τ). If $f(s^*) < f(s^{(B)})$, then, set $s^{(B)} := s^*$ and repeat Step 1.2.

- **Stage 2**.

  2.1. Set counter C to the value N. If this step has already been done once ignore the following, else set tolerance value $\tau_2 := 0.05 \cdot f(s)$ and decrement step $\Delta := 0.02$.

  2.2. Decrement C by one. Let s^* be a solution provided by the procedure NEIGHBOR(s, τ_2). If $f(s^*) < f(s^{(B)})$, then set $s^{(B)} := s^*$ and repeat 2.2.

  2.3. If $C \leq 0$ go to Step 1.2.

  2.4. Decrease the tolerance value $\tau_2 := \tau_2 - \Delta$ and reset $s := s^*$. If $\tau_2 > 0$, then go to Step 2.2.

- **Stage 3**.

  3.1. Set tolerance value $\tau := 0$.

  3.2. Let s^* be a solution provided by the procedure NEIGHBOR(s, τ). If $f(s^*) < f(s^{(B)})$, then set $s^{(B)} := s^*$ and repeat Step 3.2. Otherwise, stop and return the best found solution $s^{(B)}$.

5.2 Improved Opti-Move analysis

Results can be found in Section 8.4. With this version of the algorithm, we only slightly improved results but the behavior has changed completely for smaller sized instances. However on large instances the algorithm is limited by the efficiency of the Neighbor method. We also found by testing a large range of values that the best results are obtained with $N = 10$.

6 New heuristic

After experimentation on our improved algorithm for the Opti-Move heuristic, our supervising professor suggested a new approach to the neighbor procedure. Instead of moving the most loaded output of each round, we experimented on moving an empty output in place of the most loaded output. Unlike the previous movement, this one is always possible to make on each round.

We called this movement the ij-exclusive circular shift.

6.1 Description and Visualisation

Let's observe again the solution presented in Table 2, the new movement consist of selecting the least loaded output of the empty outputs and the most loaded of the non-empty ones. We call them q and k. For round 3 it would give 4,1. Next we take into account only the non-empty outputs between q and k and include them two as well. Again for round 3 we have 1,3,4 taken into account.

Round	Out. 1	Out. 2	Out. 3	Out 4	Out 5
3	8		9		3

Table 3: Round 3 initially

We move the empty space of output 4 to output 1. Therefore, the loaded outputs are shifting to the right by keeping the empty spaces empty.

Round	Out. 1	Out. 2	Out. 3	Out 4	Out 5
3			8	9	3

Table 4: Round 3 after the $(4, 1)$ shift

For the next step, if we look at Table 2 with the updated round 3, the least loaded empty output would be 2 and the most loaded output would be 4. The next movement therefore is to shift the empty space of output 2 into output 4. By shifting the loaded spaces circularly, we would put output 5 into output 2, and output 4 into output 5 as such:

Round	Out. 1	Out. 2	Out. 3	Out 4	Out 5
3		8	9		3

Table 5: Round 3 after the second $(2, 4)$ shift

6.2 New neighbor procedure formulation

Step 3, that deals with which outputs we shift, is changed in the new neighbor procedure.

Notations:

- $F^{(r)}(s)$ is the set of non-empty outputs in round r of solution s.
- $E^{(r)}(s)$ is the set of empty outputs in round r of solution s.

- $ij^{(q,k,r)}_{excircshift}(s)$ is the movement describes in section 6.1 between outputs q and k in round r of solution s.

Algorithm 4 ij-exclusive circular shift procedure NEIGHBOR

Input: An initial solution $s^{(0)}$ and a tolerance value τ.
Output: A potentially new solution better than $s^{(0)}$.

1. Set $i := 0$ and $r := 1$.
2. Compute $E = E^{(r)}(s^{(i)})$ and $F = F^{(r)}(s^{(i)})$.
3. If $E = \emptyset$ or $F = \emptyset$, then go to Step 5. Otherwise, choose the least loaded empty output q from E and the most loaded output k from F.
4. Let $s^{(T)}$ be a solution obtained from solution $s^{(i)}$ by doing $ij^{(q,k,r)}_{excircshift}(s^{(i)})$. If $f(s^{(T)}) < f(s^{(i)}) + \tau$, then move to a new current solution, *i.e.*, set $i := i + 1$, $s^{(i)} := s^{(T)}$.
5. If $r < |R|$, then set $r := r + 1$ and go to Step 2. Otherwise, stop and return solution $s^{(i)}$.

6.3 Iterating on the rounds

As suggested by professor Gurevsky, we experimented on a different strategy that consisted in making several movements on the same round instead of passing to the next round immediately after making a movement.

The results where not satisfying. Because the new neighborhood is too efficient, by searching in the neighborhood solution space multiple times in the same round, there is too much degradation. This can be observed in Figure 8.

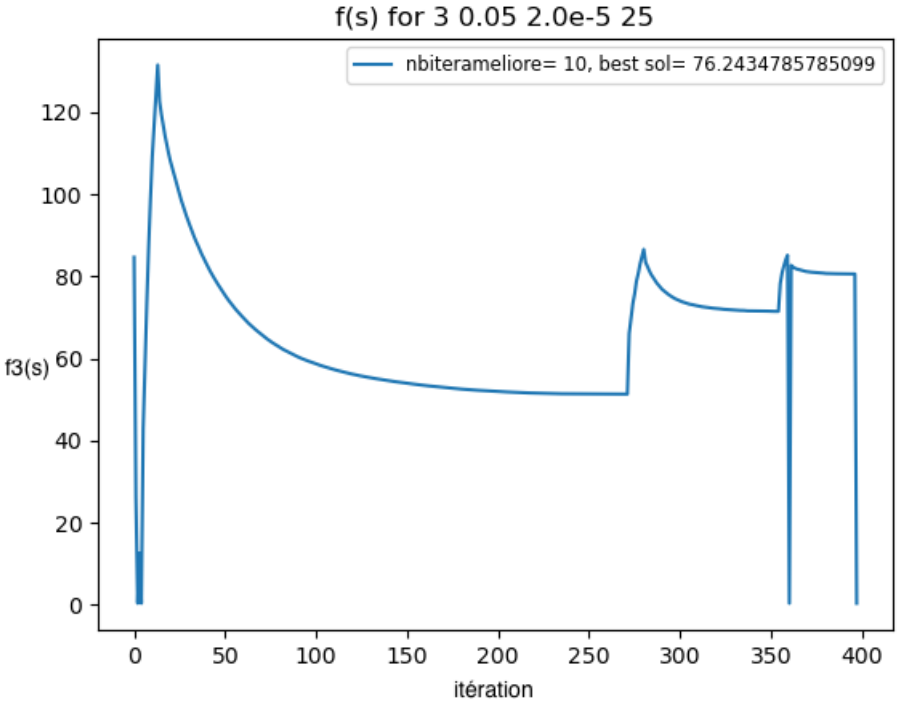

Figure 8: Evolution of solutions when iterating in the round

We have the number of iterations on the x-axis which refer to the number of calls for the neighborhood method and the value of the current solution on the y-axis. It is also the case for Figures 10, 11, 12 and 13 in Section 8.6.

6.4 New heuristic analysis

Results can be found in Section 8.5. We can observe that previous algorithms limitations have been fixed. Execution time has been heavily improved as well as the quality of the solutions, especially for the OPTICLASS instances, with an average value for f_1 at 18,18 and CPU time at 2,08 seconds.

7 Best Parameters

Throughout our experimentation, we noticed that the parameters we gave our algorithms could sometimes greatly influence the quality of the results. The parameters we wanted to check were Δ and $\mathcal{T}$. The parameter C as we observed was best most of the time when set to 10, we therefore did not experiment on it any further.

7.1 Scatter search

Our first approach was to test a number of different parameters for each instances and memorize those that gave us an optimal result. We were able to do this because at this stage, our algorithms were very efficient in time so executing all instances multiple times did not take long. To visualise the results, we made a 3D plot that showed the best parameters for Δ and $\mathcal{T}$ compared to the size of the matrix. A dot appear when the couple of Δ and *pourcentage* corresponding gave more than 15 instances of this size solved at $f_1 \leq 10$

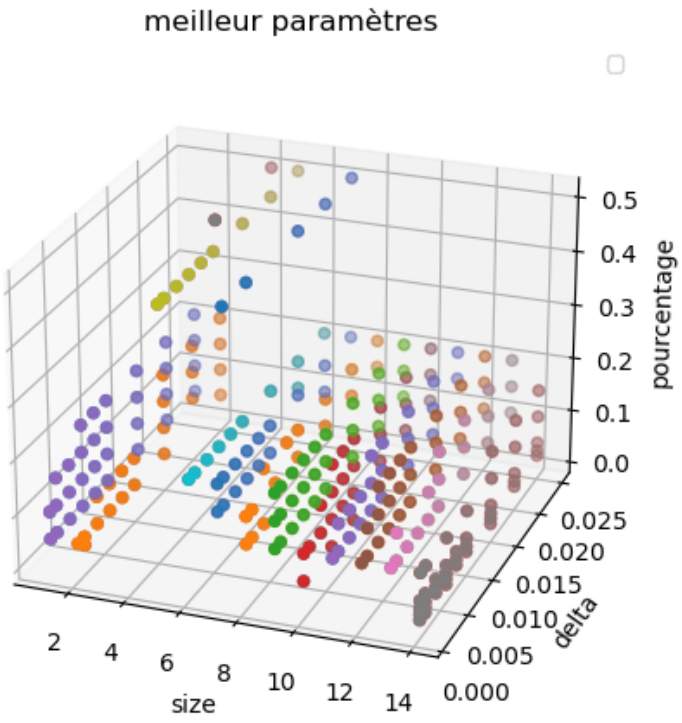

Figure 9: Scatter search results

Those results allowed to determine that on one hand the value of delta didn't influence that much the quality of solutions. On the other hand the initial value of $\mathcal{T}$ is very important but the graph didn't allow us to determine an optimal value for all instances sizes or to extract a function to compute it (Even though low values of $\mathcal{T}$ around 0.035 gave globally satisfying results)

7.2 Initialisation phase

As the previous results were insufficient we searched a more global approach to compute the the initial parameters. By looking back at the Simulated annealing metaheuristic we decided to once again take inspiration from it and implement an initialisation phase which consist of doing a determined number of randomized neighborhood (selection of q and k is randomized in new heuristic) on the initial problem and computing the standard deviation of the solutions found. Then $\mathcal{T}$ is initialized at a percentage of this value.

Algorithm 5 Initialisation

Input: An initial solution s and a number of call N.
Output: A Float V.

1. Set $C := N$, $\tau = 0$ and the list of solutions $Sol = \emptyset$.
2. Let s^* be a solution provided by the randomized procedure NEIGHBOR(s, τ).
3. Add s^* to Sol and decrement C by 1.
4. If $C = 0$ return the standard deviation of Sol. Else return to 2.

After some testing the best values are obtained when initializing τ at 1% of the obtained standard deviation which is showed in Table 14

8 Experimentation

8.1 Instances

In order to test our implementations, we were given some of the actual data from the postal industrial platforms, called OPTICLASS, and some academic instances of different sizes built for testing. There are 30 different instances for each size, and 22 OPTICLASS instances. The data came in the form of Excel sheets and we had to make a parser in order to read the data in our programs.

We had several different matrix sizes to work with, from 12x20 to 120x90. The OPTICLASS instances were the biggest in size (see Table 6).

Instances	Size
05_24_PF	204x127
05_24_PF_3_2	112x76
06_08_PF	213x214
06_09_2_PF	228x216
06_09_PF	207x213
06_16_PF	222x146
06_17_PF	204x147
06_24_PF	210x208
06_27_PF	226x202
06_27_reduit_PF	208x202
06_28_PF	204x188
06_29_PF	206x184
07_15_PF	209x196
07_15_PF_3_2	210x196
07_15_PF_3_3	111x217
07_15_PF_3_4	211x200
07_18_PF	201x124
07_19_PF	207x198
08_02_PF	205x151
08_10_PF	202x151
10_14_PF	240x211
10_27_PF_Lille	141x229

Table 6: Sizes of the OPTICLASS instances

8.2 MILP results

All of our implementations were made with the Julia programming language.

The MILP was made using JuMP, a domain-specific language for mathematical optimization [6]. The solver used was Gurobi version 10 with an academic licence. The time limit for the solver was fixed to 600 seconds.

The objective function used is f_1. As for f_2 and f_3, they were calculated after the resolution with the results of the MILP model.

The machine used has the following specifications:

Operating system macOS Monterey 12.6.3

CPU 2,7 GHz Intel Core™ i5 dual core

GPU Intel Iris Graphics 6100 1536 Mo

RAM 8 Go 1867 MHz DDR3

Instance size	#OPTI	Avg. GAP (%)	Avg. CPU (s)	Avg. f_1	Avg. f_2	Avg. f_3
12_20	30	0	3,34	9,67	3,41	3,97
20_20	30	0	3,09	10	3,54	4,08
30_15	30	0	0,91	9,33	3,28	3,81
30_30	30	0	16,36	10	3,50	4,10
40_25	30	0	7,61	9,67	3,19	3,80
40_40	29	50	97,51	10,33	3,26	3,94
50_40	30	0	113,76	10	3,26	3,91
55_55	19	89,29	381,54	59,67	15,95	19,34
60_50	25	50	334,69	11,67	4,26	4,89
75_50	24	87,5	427,26	12	4,46	5,10
80_55	0	100	NA	34,33	10,03	12,18
100_60	0	100	NA	179,33	37,34	53,83
120_90	0	100	NA	1508,67	438,10	503,79

Table 7: Results for the MILP model with f_1 as the objective function

The green cells indicate that all 30 instances have an optimal solution (which is in most cases for all instances $f_1 = 10$). The red cells indicate that none of the instances were solved within the time limit and the average GAP is at 100%. For instances of size bellow 50x40, all instances were solved to optimality. Starting from instance sizes of 80x55, no instances were solved to optimality under the time limit of 600 seconds.

These results were to be expected for large data.

8.3 Opti-Move results

The following results were obtained with a machine having these specifications:

Operating system Windows 11 22H2

CPU Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz 3.19 GHz

GPU Nvidia Geforce GTX 1080

RAM 32 Go 2666 GHz DDR4

These specifications were used throughout the rest of our experimentation. The objective function used to optimize from now on is f_3 and as for f_1 and f_2, they were calculated after resolution

Instance	Avg. f_1	Avg. f_2	Avg. f_3	Avg. CPU (s)
12x20	95,67	23,36	27,78	0,00
20x20	39,33	7,86	10,30	0,01
30x15	33,33	7,92	9,85	0,01
30x30	74,00	13,01	17,21	0,02
40x25	50,33	9,29	12,16	0,02
40x40	74,33	15,55	18,93	0,04
50x40	58,33	9,72	12,63	0,05
55x55	71,67	11,50	15,52	0,14
60x50	67,00	10,76	14,09	0,10
75x50	70,00	11,33	14,93	0,12
80x55	59,33	9,31	12,14	0,19
100x60	72,00	11,19	15,38	0,28
120x90	54,67	7,71	10,01	1,02
OPTICLASS	1002,27	205,17	244,36	7,79

Table 8: Opti-Move heuristic average results

Instance	f_1	f_2	f_3	CPU (s)
05_24_PF	50	6,57	7,82	2,13
05_24_PF_3_2	30	6,37	7,45	0,40
06_08_PF	1850	419,83	491,64	4,87
06_09_2_PF	2440	532,34	625,89	4,64
06_09_PF	1980	475,52	544,10	12,50
06_16_PF	50	6,12	7,74	5,37
06_17_PF	60	4,44	7,89	5,27
06_24_PF	1680	456,15	505,31	11,15
06_27_PF	1630	294,09	354,58	5,34
06_27_reduit_PF	1470	254,58	341,46	7,89
06_28_PF	50	3,16	6,27	7,47
06_29_PF	30	2,45	4,89	6,51
07_15_PF	310	54,69	84,77	51,38
07_15_PF_3_2	670	169,25	218,37	10,96
07_15_PF_3_3	1430	253,13	295,90	0,72
07_15_PF_3_4	1000	224,45	288,12	9,10
07_18_PF	60	6,73	8,56	1,56
07_19_PF	50	6,58	8,13	10,63
08_02_PF	80	6,25	10,95	2,68
08_10_PF	60	6,53	9,72	3,60
10_14_PF	5000	939,01	1095,34	6,68
10_27_PF_Lille	2070	385,55	450,92	0,51

Table 9: Opti-Move heuristic results for OPTICLASS

8.4 Improved Opti-Move results

Instance	Avg. f_1	Avg. f_2	Avg. f_3	Avg. CPU (s)
12x20	86,33	21,27	25,29	0,00
20x20	41,67	7,77	10,42	0,01
30x15	36,67	8,55	10,92	0,01
30x30	70,33	12,26	16,82	0,02
40x25	44,67	8,40	11,12	0,02
40x40	58,33	11,74	14,63	0,05
50x40	41,33	7,15	9,31	0,06
55x55	37,67	6,33	8,25	0,17
60x50	48,00	8,11	10,93	0,13
75x50	53,00	9,25	12,81	0,14
80x55	33,67	6,17	7,78	0,25
100x60	47,67	8,83	11,90	0,34
120x90	29,00	5,51	6,85	1,36
OPTICLASS	975,00	199,11	237,10	9,21

Table 10: Improved Opti-Move average results

Instance	f_1	f_2	f_3	CPU (s)
05_24_PF	30	5,14	5,30	2,22
05_24_PF_3_2	20	5,10	5,40	0,50
06_08_PF	1730	401,30	471,25	15,46
06_09_2_PF	2490	533,21	629,80	4,12
06_09_PF	1980	509,36	574,93	12,93
06_16_PF	20	4,15	4,93	7,09
06_17_PF	20	2,57	4,16	6,78
06_24_PF	1750	455,47	510,63	13,12
06_27_PF	1590	291,19	352,14	6,38
06_27_reduit_PF	1430	253,03	340,53	44,97
06_28_PF	30	2,85	5,55	9,81
06_29_PF	20	1,28	3,29	9,19
07_15_PF	420	85,20	122,70	11,96
07_15_PF_3_2	340	63,29	93,33	13,29
07_15_PF_3_3	1270	253,05	292,88	3,56
07_15_PF_3_4	670	171,59	218,77	12,18
07_18_PF	70	5,60	7,46	2,46
07_19_PF	70	6,09	7,69	13,41
08_02_PF	50	3,23	6,46	3,66
08_10_PF	50	3,67	6,12	4,53
10_14_PF	5340	938,66	1102,10	4,16
10_27_PF_Lille	2060	385,39	450,71	0,75

Table 11: Improved Opti-Move results for OPTICLASS

8.5 New version heuristic results

Instance	Avg. f_1	Avg. f_2	Avg. f_3	Avg. CPU (s)
12x20	20,67	5,12	6,31	0,01
20x20	18,33	4,55	5,49	0,03
30x15	16,00	4,27	5,24	0,03
30x30	20,33	4,77	5,83	0,05
40x25	17,67	4,11	5,13	0,06
40x40	19,67	4,41	5,54	0,08
50x40	18,33	4,16	5,18	0,10
55x55	18,67	4,12	5,02	0,13
60x50	20,00	4,37	5,41	0,15
75x50	18,33	4,30	5,25	0,18
80x55	20,33	4,34	5,32	0,23
100x60	17,00	3,54	4,50	0,28
120x90	19,33	4,55	5,45	0,46
OPTICLASS	18,18	3,70	4,55	2,08

Table 12: New heuristic average results with initialisation

Instance	f_1	f_2	f_3	CPU (s)
05_24_PF	10	4,98	4,99	0,72
05_24_PF_3_2	10	4,78	4,89	0,22
06_08_PF	20	4,69	4,91	5,35
06_09_2_PF	10	4,24	4,61	4,52
06_09_PF	20	3,18	4,26	2,75
06_16_PF	10	4,04	4,49	1,05
06_17_PF	10	1,83	3,03	1,63
06_24_PF	10	4,61	4,80	2,42
06_27_PF	10	2,60	3,60	2,81
06_27_reduit_PF	20	4,50	4,81	1,63
06_28_PF	20	1,05	3,09	1,60
06_29_PF	20	1,06	2,94	1,61
07_15_PF	20	5,05	5,10	1,41
07_15_PF_3_2	20	4,71	5,25	1,55
07_15_PF_3_3	30	1,84	3,25	1,75
07_15_PF_3_4	20	4,94	5,36	1,76
07_18_PF	10	4,67	4,83	0,74
07_19_PF	30	4,72	5,27	2,43
08_02_PF	40	1,49	3,91	1,26
08_10_PF	10	2,20	3,32	1,71
10_14_PF	30	7,45	8,69	4,84
10_27_PF_Lille	20	2,82	4,70	1,95

Table 13: New heuristic results for OPTICLASS with initialisation

8.5.1 1 percent S.D. (Standard Deviation) in initialisation

Instance	Avg. f_1	Avg. f_2	Avg. f_3	Avg. CPU (s)
12x20	37,67	8,82	10,72	0,01
20x20	20,33	5,13	6,11	0,04
30x15	10,00	3,36	3,94	0,29
30x30	14,67	3,97	4,77	0,09
40x25	10,33	3,27	3,93	0,27
40x40	13,33	3,39	4,25	0,11
50x40	10,33	3,29	3,96	0,22
55x55	14,00	3,67	4,44	0,15
60x50	12,33	3,40	4,10	0,30
75x50	11,67	3,42	4,04	0,55
80x55	12,67	3,61	4,33	0,60
100x60	10,33	3,09	3,83	0,76
120x90	12,67	3,63	4,26	0,86
OPTICLASS	12,73	3,37	4,00	4,52

Table 14: New heuristic average results with 1 percent S.D. in the initialisation

Instance	f_1	f_2	f_3	CPU (s)
05_24_PF	10	4,98	4,99	1,39
05_24_PF_3_2	10	4,78	4,89	0,26
06_08_PF	10	4,63	4,81	1,83
06_09_2_PF	10	4,24	4,61	2,33
06_09_PF	10	2,87	3,79	1,99
06_16_PF	10	4,04	4,49	1,01
06_17_PF	10	1,83	3,03	0,83
06_24_PF	10	4,61	4,80	1,18
06_27_PF	10	2,60	3,60	1,26
06_27_reduit_PF	10	4,43	4,71	1,06
06_28_PF	10	0,21	1,03	1,10
06_29_PF	10	0,43	1,46	0,88
07_15_PF	10	5,00	5,00	7,37
07_15_PF_3_2	10	4,29	4,63	8,05
07_15_PF_3_3	10	1,75	2,96	1,35
07_15_PF_3_4	20	4,62	4,87	7,99
07_18_PF	10	4,67	4,83	11,57
07_19_PF	20	4,37	4,76	8,43
08_02_PF	40	1,49	3,91	16,49
08_10_PF	10	2,20	3,32	12,25
10_14_PF	10	4,50	4,74	8,58
10_27_PF_Lille	20	1,53	2,85	2,23

Table 15: New heuristic results with 1 percent S.D. at initialisation for OPTICLASS

8.6 Plots

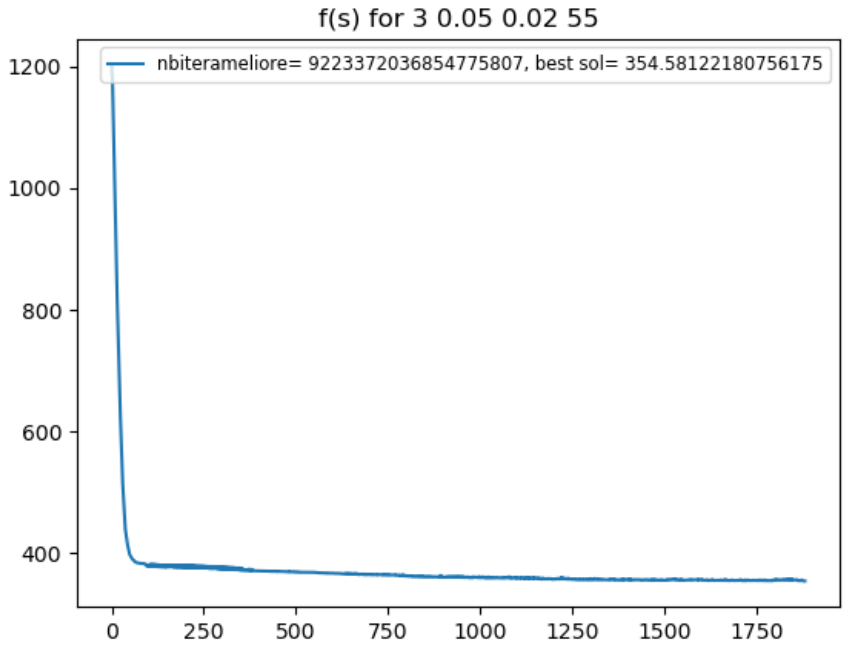

Figure 10: Evolution of solutions for instance 06_27_PF with Opti-Move heuristic

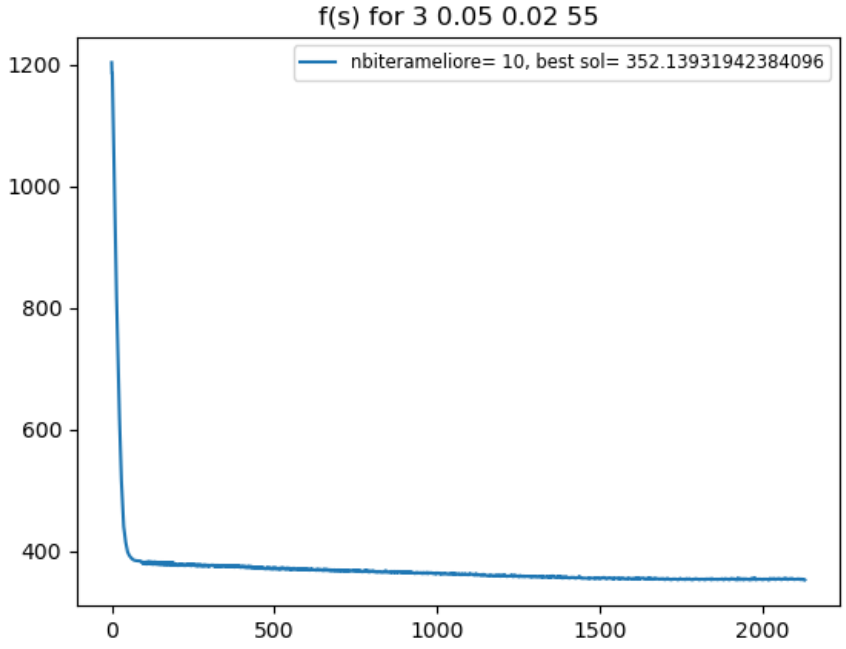

Figure 11: Evolution of solutions for instance 06_27_PF with Improved Opti-Move heuristic

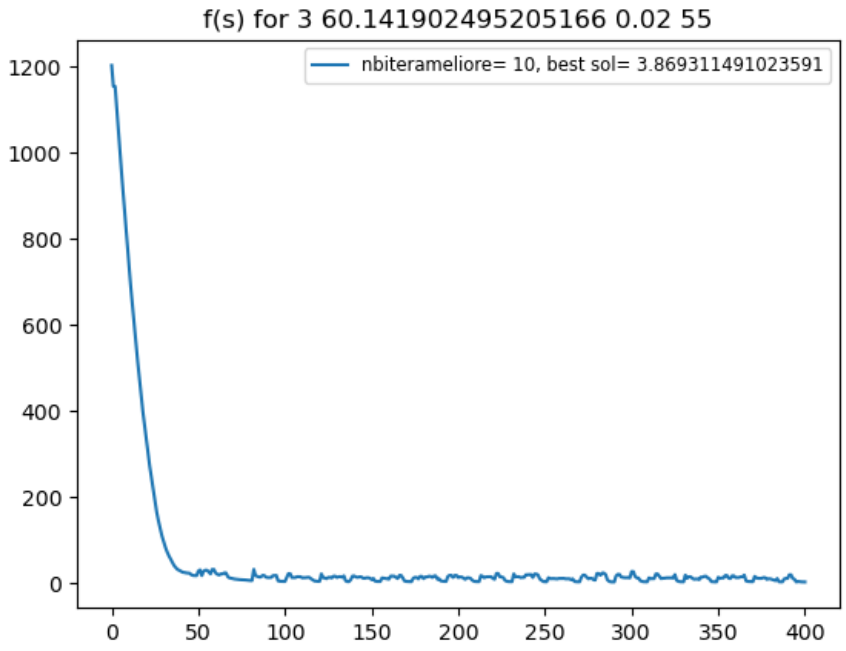

Figure 12: Evolution of solutions for instance 06_27_PF with New heuristic

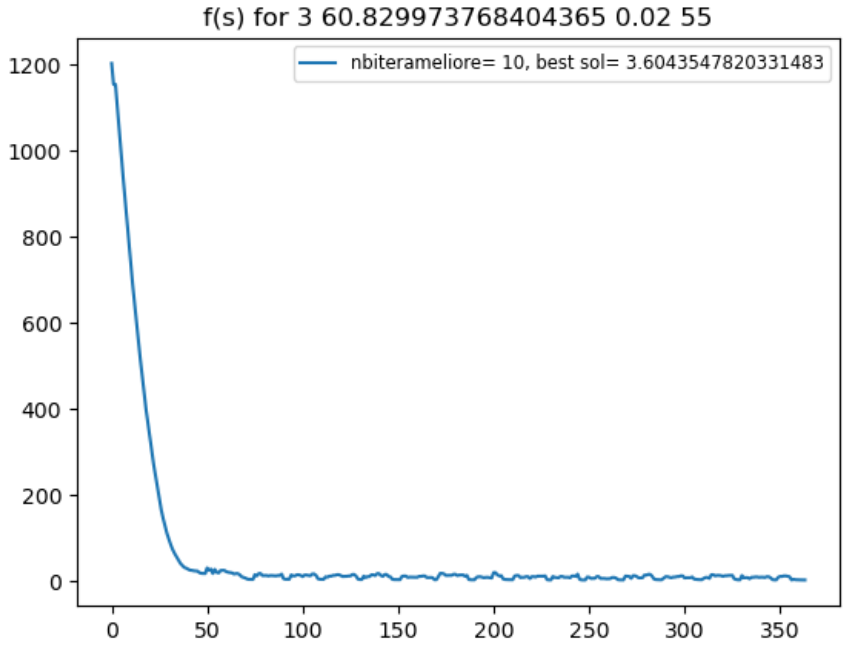

Figure 13: Evolution of solutions for instance 06_27_PF with New heuristic and with 1% standard deviation at initialisation

8.7 Comparison and discussion around the results

To compare the results, we are focusing on the value of f_1, which is according to us the most easy to apprehend of the fitness functions.

Instance	Opti-Move	Improved Opti-Move	New heuristic	New heuristic with 1% s.d.
05_24_PF	50	30	10	10
05_24_PF_3_2	30	20	10	10
06_08_PF	1850	1730	20	10
06_09_2_PF	2440	2490	10	10
06_09_PF	1980	1980	20	10
06_16_PF	50	20	10	10
06_17_PF	60	20	10	10
06_24_PF	1680	1750	10	10
06_27_PF	1630	1590	10	10
06_27_reduit_PF	1470	1430	20	10
06_28_PF	50	30	20	10
06_29_PF	30	20	20	10
07_15_PF	310	420	20	10
07_15_PF_3_2	670	340	20	10
07_15_PF_3_3	1430	1270	30	10
07_15_PF_3_4	1000	670	20	20
07_18_PF	60	70	10	10
07_19_PF	50	70	30	20
08_02_PF	80	50	40	40
08_10_PF	60	50	10	10
10_14_PF	5000	5340	30	10
10_27_PF_Lille	2070	2060	20	20

Table 16: OPTICLASS results for f_1

Instance	Opti-Move	Improved Opti-Move	New heuristic	New heuristic with 1% s.d.
12x20	95,67	86,33	20,67	37,67
20x20	39,33	41,67	18,33	20,33
30x15	33,33	36,67	16,00	10,00
30x30	74,00	70,33	20,33	14,67
40x25	50,33	44,67	17,67	10,33
40x40	74,33	58,33	19,67	13,33
50x40	58,33	41,33	18,33	10,33
55x55	71,67	37,67	18,67	14,00
60x50	67,00	48,00	20,00	12,33
75x50	70,00	53,00	18,33	11,67
80x55	59,33	33,67	20,33	12,67
100x60	72,00	47,67	17,00	10,33
120x90	54,67	29,00	19,33	12,67
OPTICLASS	1002,27	975,00	18,18	12,73

Table 17: Average results for f_1

We can observe that throughout our implementations, we managed to drastically lower the average values of the instances. We observe a degradation of average results for instance sizes 12x20 and 20x20. As our main goal is to optimize real size data, this is to be expected.

We are satisfied by our OPTICLASS results which was the main focus of our research.

9 Conclusion

We reached all goals that where set for our research project. Our implementations work on all instances and generates near optimal results with short execution times.

What we liked most about this project is that it was a real case study about an industrial problem that faced a french company and not a theoretical research project. It allowed us to observe the organization of mail sorting in a national mail company. We had the opportunity to link our theoretical knowledge to real world observation by visiting the industrial mail platform of La Poste near Rennes, which was a welcomed experience. Even though mail sorting was not especially a very engaging subject at first glance, we eventually appreciated working on it and became very involved in the research.

The most difficult part was to become familiar with the problem and its notations. We had to be ingenious about making an efficient implementation of the MILP and we had to be meticulous about the functioning of the heuristic proposed by Professor Gurevsky. However, we had great amount of help understanding the context with the previous work of Emmanuelle Amann's internship and ongoing thesis.

The research allowed us to improve our knowledge about heuristics and in particular how to tailor an existing algorithm and adapt it to a given problem in regards of the optimization needs. It also taught us how to carry out a more significant scientific research, keeping logs of our results and making improvements towards a set goal.

10 Acknowledgements

We would like to thank our supervising Professor Evgeny Gurevsky for his guidance, counseling and advices throughout our research. We would also like to thank Emmanuelle Amann for taking her time to explain the context of the mail sorting at La Poste and letting us use some of her illustrative figures.

References

[1] Emmmanuelle Amann. *Optimisation des plans de tri courrier de La Poste*. Sorbonne Université, 2022.

[2] M. E. Salveson. "The Assembly-Line Balancing Problem". In: *Journal of Fluids Engineering* 77.6 (Aug. 1955), pp. 939–947. DOI: 10.1115/1.4014559. URL: https://doi.org/10.1115/1.4014559.

[3] Peter J. M. van Laarhoven and Emile H. L. Aarts. *Simulated Annealing: Theory and Applications*. Springer Netherlands, 1987. DOI: 10.1007/978-94-015-7744-1. URL: https://doi.org/10.1007/978-94-015-7744-1.

[4] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. "Optimization by Simulated Annealing". In: *Science* 220.4598 (1983), pp. 671–680. DOI: 10.1126/science.220.4598.671. eprint: https://www.science.org/doi/pdf/10.1126/science.220.4598.671. URL: https://www.science.org/doi/abs/10.1126/science.220.4598.671.

[5] Evgeny Gurevsky Emmanuelle Amann and Nasser Mebarki. "Mail sorting optimization: a case study of the french postal company La Poste". Ifac World Congress. 2023.

[6] *JuMP*. URL: https://jump.dev/JuMP.jl/stable/ (visited on 05/11/2023).