# MULTI-OBJECTIVE COST OPTIMIZATION IN PI CROSS-DOCK NETWORKS USING GENETIC ALGORITHMS

**Master 2 Réseaux et système autonomes, paris cité**

Presented by Zineb Taieb

Supervisors:
Mme Essghaier Fatma
M Allaoui Hamid

Motivation & context →

# WHY GA FOR CROSS DOCK OPTIMIZATION?

- **Objectif**

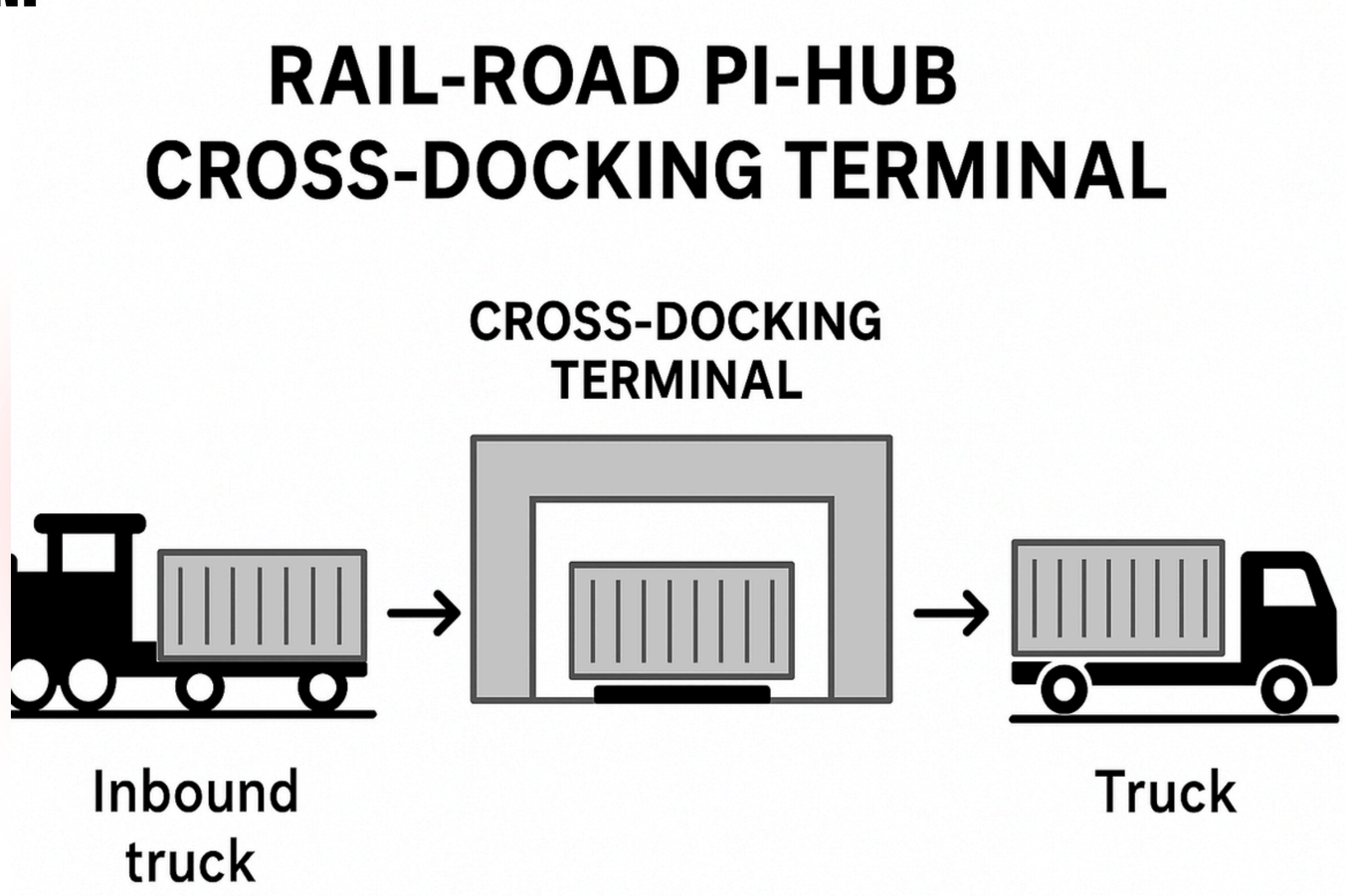
  We aim to optimize container-truck assignments to minimize transport cost and energy cost

- **NP-hard**

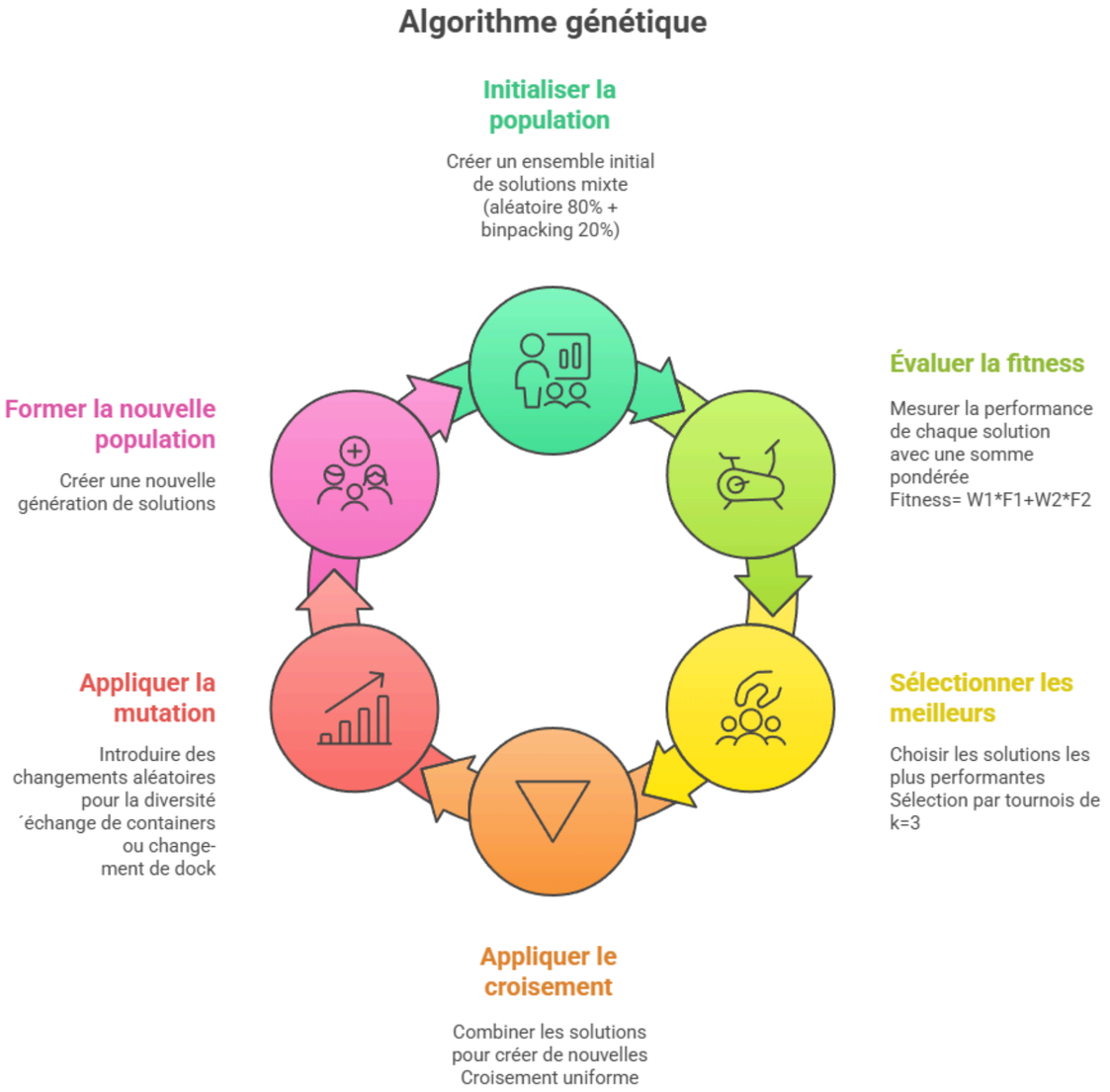
  Optimization of this kind of problems is NP-hard

- **Why Genetic algorithms?**

  Robustness,  Adaptability, Flexibility(hybridization)



# GENETIC ALGORITHM PRINCIPAL



# CHROMOSOME DESIGN

| Containers | 0 | Dock position | 0 | Containers | 0 | Dock position | 0 | Containers | 0 | Dock position | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| [4] | 0 | 1 | 0 | [] | 0 | 2 | 0 | [1,2] | 0 | 4 | 0 |

| Containers | 0 | Dock position | 0 | Containers | 0 | Dock position | 0 |
|---|---|---|---|---|---|---|---|
| [3,6] | 0 | 3 | 0 | [] | 0 | 5 | 0 |

- A  chromosome represents a solution
- Each block of 4 represents a truck and its assignements

# FITNESS FUNCTION

- Shows how to evaluate each solution
- Fitness function = Objective function

$$\text{Minimize } F_1 = \sum_{h=1}^{H} \sum_{d=1}^{D} CT_d \, a_{hd} \text{Minimize } F_2 = CE \sum_{i=1}^{N} \sum_{h=1}^{H} z_{ih}$$

- Weighted sum of F1 and F2 , with weights W1, W2

Fitness = W1* F1 + W2 * F2 ; W1=0.5 and W2=0.5

# GENETIC OPERATIONS

## Crossover

Flip a coin :
if random.random()< crossover_rate=0.9
Uniforme crossover

Parent 1 : [ [4,6],0,3,0, {Point de croisement}[1],0, 5 ,0,[],0, 1 ,0 ]

Parent 2 : [ [3,7],0,4 ,0,{Point de croisement} [],0,7 ,0,[2],0, 6, 0 ]

Enfant 1 : [ [4,6],0,3,0, [],0,7 ,0,[2],0, 6, 0 ]

Enfant 2 : [ [3,7],0,4 ,0, [1],0, 5 ,0,[],0, 1 ,0 ]

Else:
Enfant 1 =parent 1
Enfant 2 =parent 2

## Mutation

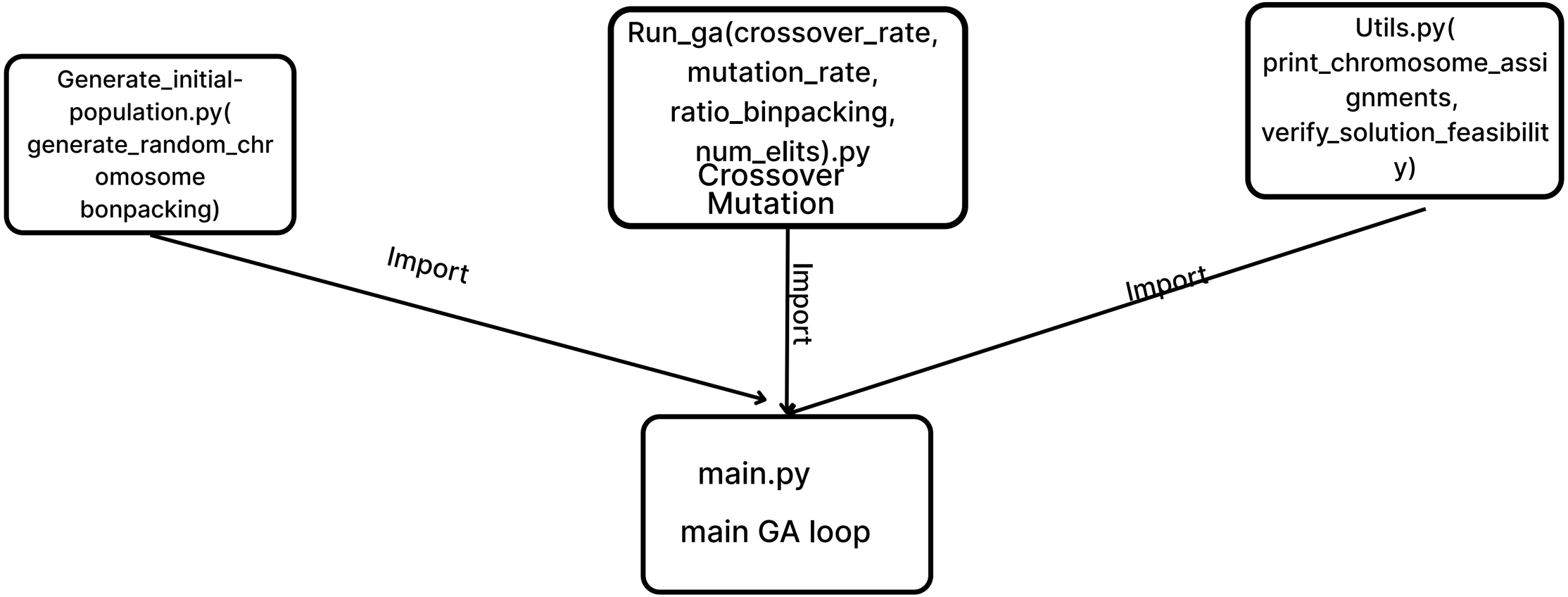
Chromosome : [ [4,6],0,3,0,[],0,7 ,0,[2],0, 6, 0 ]

Flip a coin :
if random.random()< mutation_rate=0.03:

Mutate dock position
Chromosome : [ [4,6],0,5,0,[],0,3 ,0,[2],0, 1, 0 ]

Swap containers
Chromosome: [ [2],0,5,0,[],0,3 ,0,[4,6],0, 1, 0 ]

# IMPLEMENTATIONS DETAILS

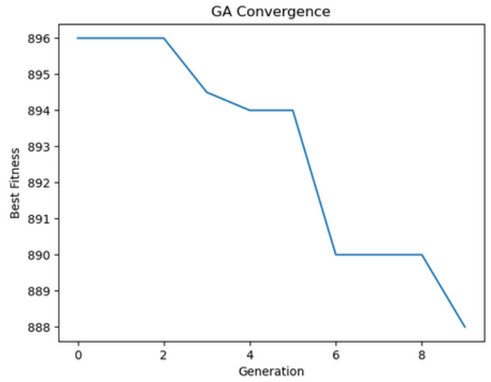



| Crossover_rate | Mutation_rate | Ratio_binpacking | num_elits |
|---|---|---|---|
| 0.9 | 0.03 | 0.8 | 1 |

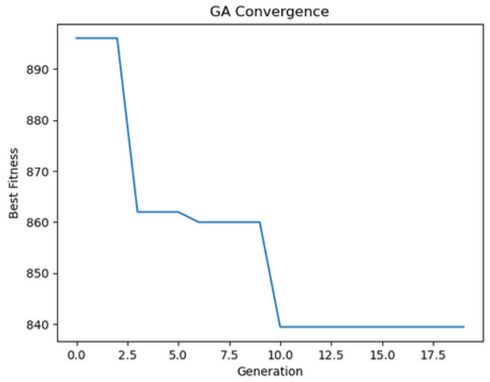
# Convegence of the fitness evaluation

| Experience | Population | Generations | temps d'execution (secon des) | Valeur Fitness |
|---|---|---|---|---|
| A | 10 | 10 | 58.405 | 888.0 |
| B | 30 | 20 | 52.034 | 839.5 |
| C | 100 | 50 | 92.124 | 832.5 |

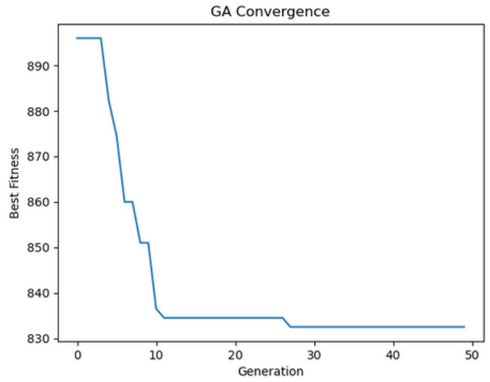
Experience A




Experience B




Experience C



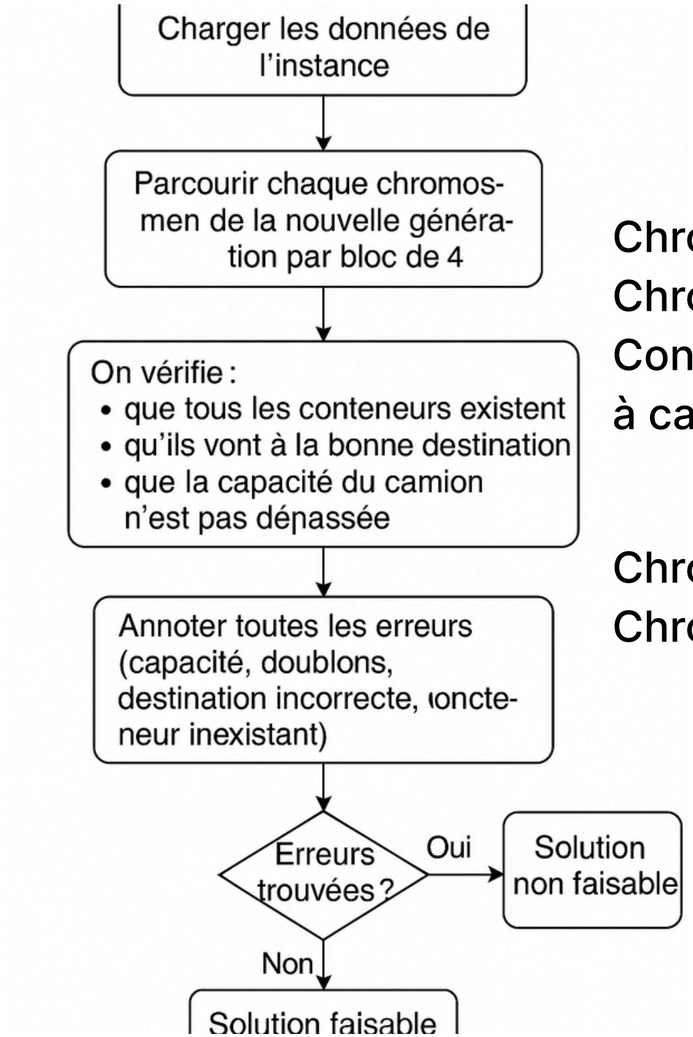

More population = more diversity

# Convegence of the fitness evaluation

| Mutation rate | Best fitnessue | Observation |
|---|---|---|
| 0.05 | 880 | Bonne convergence, mais un peu lente |
| 0.1 | 881 | Convergence plus rapide, mais pas plus optimale |
| 0.07 | **877** | Meilleur équilibre : meilleure solution finale |

# Feasible and unfeasible solutions

Feasable solution = solution that respects constraints ( truck capacity, destination and assignement)




Génération 1 :

Nouvelle population

Chromosome 1: [[], 0, 4, 0, [6, 3], 0, 4, 0, [1, 2], 0, 4, 0, [], 0, 5, 0, [4], 0, 3, 0]

Chromosome 1 : Non faisable

Conteneur 6 (dest 2) assigné à camion 2 (dest 1) Conteneur 3 (dest 2) assigné à camion 2 (dest 1) Conteneur 4 (dest 1) assigné à camion 5 (dest 2)

Chromosome 5: [[], 0, 5, 0, [], 0, 3, 0, [1, 2], 0, 2, 0, [3, 6], 0, 4, 0, [], 0, 5, 0]

Chromosome 5 : Faisable

# Handling constraints and unfeasible solutions

- Rejection strategy
- Repair strategy
- Penalize strategy

Penalize strategy ✅

Fitness = W1* F1 + W2 * F2 + penalty

Problem with simple penalty

The GA minimizes : Fitness = W1* F1 + W2 * F2 + penalty

The GA found :

A chromosome with fewer containers → minimizes the total fitness

Penalty is small, GA prefers minimizing total cost over penalty.

Example output

```
Meilleur Chromosome trouvé : [[], 0, 5, 0, [], 0, 5, 0, [3, 1], 0, 4, 0, [6], 0, 5, 0, [], 0, 5, 0]
```

Only 3 containers assigned out of 6, Penalty is weak

# Handling constraints and unfisable solutions

Adaptative Penalizing strategy:

- Penalize every container assigned more than once in the same chromosome
- Penalize every container assigned to a truck with different destination
- Penalize if truck capacity is exceeded
- Penalize every unassigned container

```
P_DUP = 500 #penalty for duplicated container in the chromosome.
P_CAP = 200 # penalty for exceeded capacity
P_DEST = 300 #penalty for wrong destination
P_UNASSIGNED = 5000 #penalize heavely  for unassigned container
assigned_containers_list []
for i in assigned_container_list:
   if count_assined[i]>1:
       Fitness (chromosome)= fitness + P_DUP
if truck_capacity< container length:
   fitness (chromosome) = fitness + P_CAP
unassigned_list []
n_b_unassigned= len(unassigned_list )
fitness ( chromosome) = fitness + P_UNASSIGNED * n_b_unassigned #penalize every truck unassigned in the
 chromosome
```

Output of adaptative penalizing stategy

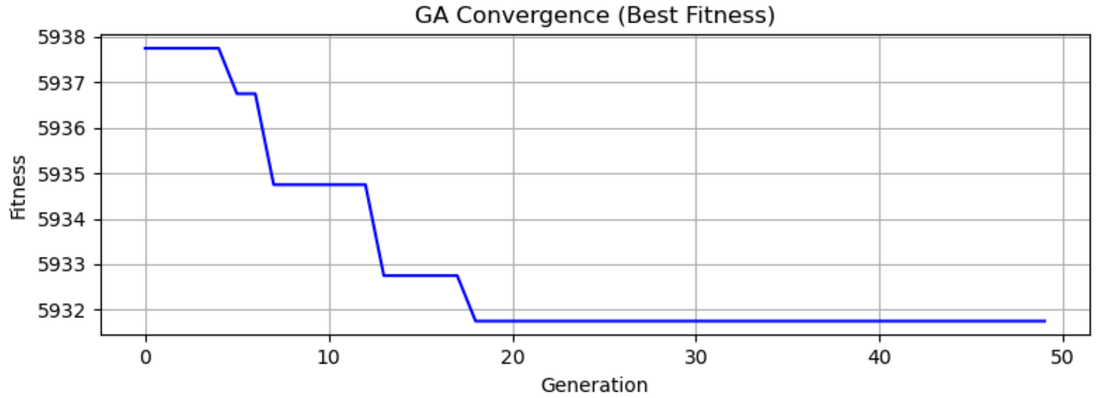
```
Meilleur Chromosome trouvé : [[4], 0, 5, 0, [], 0, 2, 0, [1, 2], 0, 5, 0, [3, 6], 0, 4, 0, [], 0, 2, 0]
```

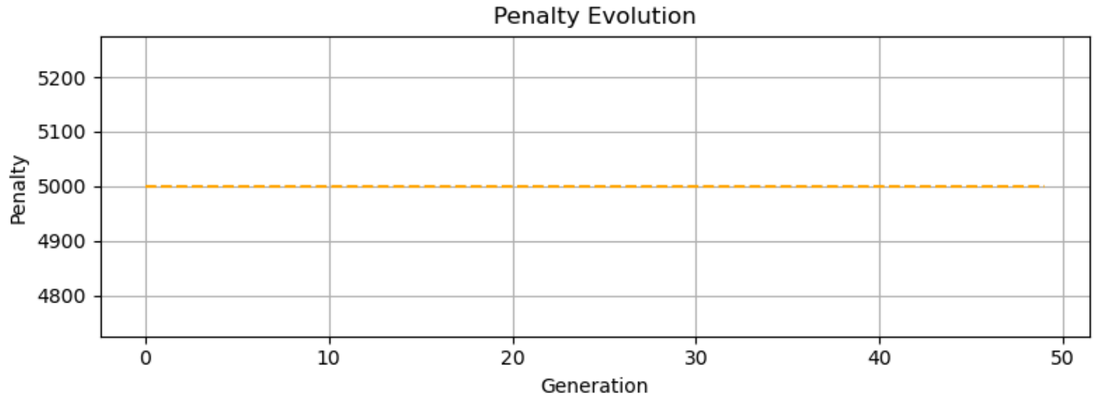
5 containers assigned, 1 unassigned container

# Results and discussions

Fitness convergence drops over the generations → normal

Penalty evolution over the generations stays flat = 5000→ the GA never get better with penalty, same penalty over the generations




**Root cause:**

- Check Initial population:

```
=== Génération 1 : Nouvelle population ===
Chromosome 1:
[[4], 0, 4, 0, [], 0, 3, 0, [1, 2], 0, 3, 0, [3, 6], 0, 4, 0, [], 0, 5, 0]

Chromosome 1 :  Non faisable
    ⚠️ Conteneurs non assignés : [5]
.
.
.
=== Génération 50 : Nouvelle population ===
Chromosome 1:
[[4], 0, 4, 0, [], 0, 3, 0, [1, 2], 0, 3, 0, [3, 6], 0, 4, 0, [], 0, 5, 0]
Chromosome 1 :  Non faisable
    ⚠️ Conteneurs non assignés : [5]
```

→Container 5 never gets assigned

→Checking input data, Container[5].length = 10 > max truck capacity = 6

# Results and discussions

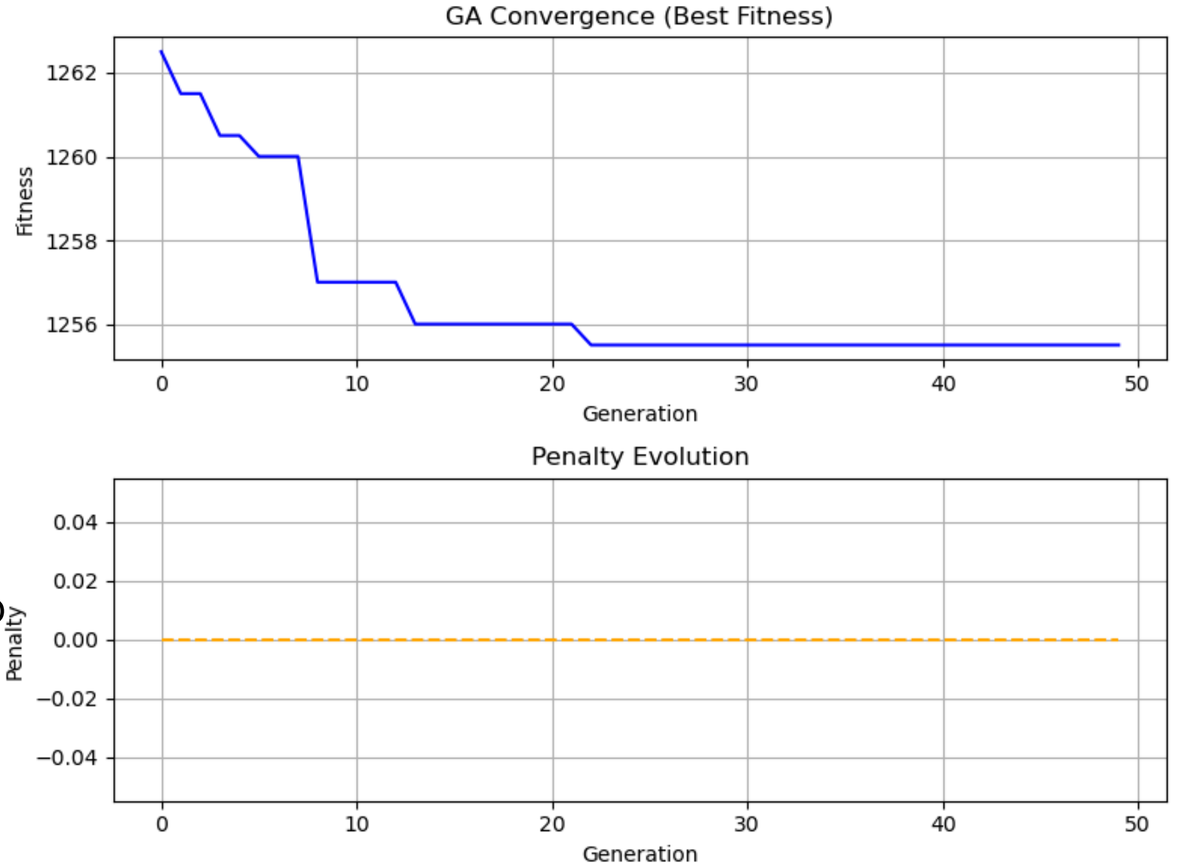
- After reinitialization:
- Container[5].length = 5

**Results**

Fitness curve (top) : → Gradually decreases then stabilizes around generation 21.

Penalty curve (bottom) : → Perfectly flat at 0.

-->Meaning : every solution is feasible, no unassigned containers, no wrong destination and no capacity exceeded.



# Exact Method

GUROBI OPTIMIZATION

Instances param

Container_lengths[1, 2, 3, 4, 5]
Trucks size = 6
Cost_truck_by_dest
Cost_one energy_unit= 0,5

Optimize

$$\text{Minimize } F_1 = \sum_{h=1}^{H} \sum_{d=1}^{D} CT_d \, a_{hd} \text{Minimize } F_2 = CE \sum_{i=1}^{N} \sum_{h=1}^{H} z_{ih}$$

Unified Objective function: Weighted Sum, W1=0.5  W2= 0.5

$$Minimize \, (\mathrm{W1} \times F_1 \times W2 \times F_2)$$

Gurobi License : Free academic license, version 12

## Instances

| Instance | CE(cost of 1 truck) | D(destination) | H (Trucks) | K(Docks) | N(Containers) |
|---|---|---|---|---|---|
| 1 | 0.5 | 2 | 10 | 5 | 4 |
| 2 | 0.5 | 3 | 9 | 5 | 8 |
| 3 | 0.5 | 4 | 10 | 7 | 5 |
| 4 | 0.5 | 3 | 7 | 3 | 11 |
| 5 | 0.5 | 3 | 10 | 4 | 13 |
| 6 | 0.5 | 3 | 9 | 5 | 11 |
| 7 | 0.5 | 2 | 9 | 3 | 13 |
| 8 | 0.5 | 3 | 10 | 5 | 11 |
| 9 | 0.5 | 5 | 6 | 4 | 9 |
| 10 | 0.5 | 4 | 5 | 7 | 10 |
| 11 | 0.5 | 5 | 6 | 7 | 7 |
| 12 | 0.5 | 3 | 7 | 3 | 9 |
| 13 | 0.5 | 5 | 7 | 5 | 5 |
| 14 | 0.5 | 5 | 8 | 7 | 10 |
| 15 | 0.5 | 3 | 5 | 4 | 6 |
| 16 | 0.5 | 2 | 10 | 5 | 6 |
| 17 | 0.5 | 3 | 6 | 5 | 4 |
| 18 | 0.5 | 5 | 10 | 5 | 11 |

# GA vs method_exact

Gap(%) = { 100 × (Couts_GA – Couts_Optimals)Couts_optimals, si Couts_GA > Couts_optimals

0, sinon

– Unsolved cases by Gurobi: 9, 10, 12, 13, 19, 24, 25, 26, 29 (status 3, infeasible).
– If Couts_GA> Couts_optimals, we measure the gap between Couts_GA and the optimal solution.
– If Couts_GA < Couts_optimals, Gap = 0, meaning the genetic algorithm found a more optimal solution than the exact method.

- Execution time: for some instances, especially the larger ones, the exact method was very slow (up to 1056 seconds).
- Ga with random initialization, significantly less stable, this indicates strong dependence of the GA on the initialization.

| Instance | Couts_GA | Couts_GA_random | Couts_Exact | Gap(%) | ExecutionTime(s)_GA | ExecutionTime(s)_random | ExecutionTime(s)_xact |
|---|---|---|---|---|---|---|---|
| 1 | 2046 | 1230,25 | 1439,0016 | 42,18191 | 3,818 | 0,411 | 0,142 |
| 2 | 4673,25 | 2347,5 | 2888,0014 | 61,81606 | 5,025 | 0,431 | 0,079 |
| 3 | 1278,25 | 11556 | 721,0008 | 77,28829 | 3,939 | 0,527 | 0,086 |
| 4 | 2832,5 | 17722,75 | 1555.500499 | 82,09573 | 4,669 | 0,774 | 1,475 |
| 5 | 6094 | 9362,75 | 3643,5017 | 67,25668 | 6,76 | 0,6 | 208,119 |
| 6 | 3809 | 17909,5 | 2347,5015 | 62,25762 | 4,953 | 0,867 | 256,243 |
| 7 | 4897,5 | 5240,5 | 2582,0007 | 89,67849 | 5,959 | 0,805 | 0,338 |
| 8 | 3384 | 2479 | 1874,0035 | 80,57597 | 6,573 | 0,341 | 311,724 |
| 11 | 1185,75 | 8066 | 1304,0004 | 0 | 2,926 | 0,747 | 0,209 |
| 14 | 2177,5 | 5036,75 | 1375,0006 | 58,36357 | 5,632 | 0,537 | 22,491 |
| 15 | 1867 | 8488,75 | 1340,5012 | 39,27626 | 2,437 | 0,627 | 0,026 |
| 16 | 981,75 | 4708,25 | 588,5008 | 66,8222 | 4,842 | 0,442 | 0,072 |
| 17 | 1359 | 3559,25 | 745,001 | 82,41586 | 2,747 | 1,028 | 0,029 |
| 18 | 5493,5 | 11578 | 2533,0048 | 116,8768 | 6,201 | 0,593 | 17,468 |
| 20 | 5772,5 | 13756,25 | 3325,002 | 73,60892 | 7,34 | 0,87 | 1059,178 |
| 21 | 2384,25 | 9647,25 | 1691,002 | 40,99629 | 4,13 | 0,76 | 0,129 |
| 22 | 1555,5 | 11172 | 831,002 | 87,18367 | 3,368 | 0,813 | 0,043 |
| 23 | 1946,75 | 2552,25 | 1070,001 | 81,93908 | 4,38 | 0,415 | 0,248 |
| 27 | 2553,25 | 1309,5 | 1367,0017 | 86,77738 | 4,098 | 0,494 | 17,248 |
| 28 | 1922,25 | 1855,5 | 1056,5006 | 81,945 | 3,542 | 0,262 | 0,159 |
| 30 | 2305 | 5715,5 | 1306,0024 | 76,49278 | 6,813 | 0,556 | 0,108 |

# Instance 9 , noted unfeasible by Gurobi, let's see why?

| Destinations | Trucks | Docks | Containers |
|---|---|---|---|
| 5 | 6 | 4 | 9 |

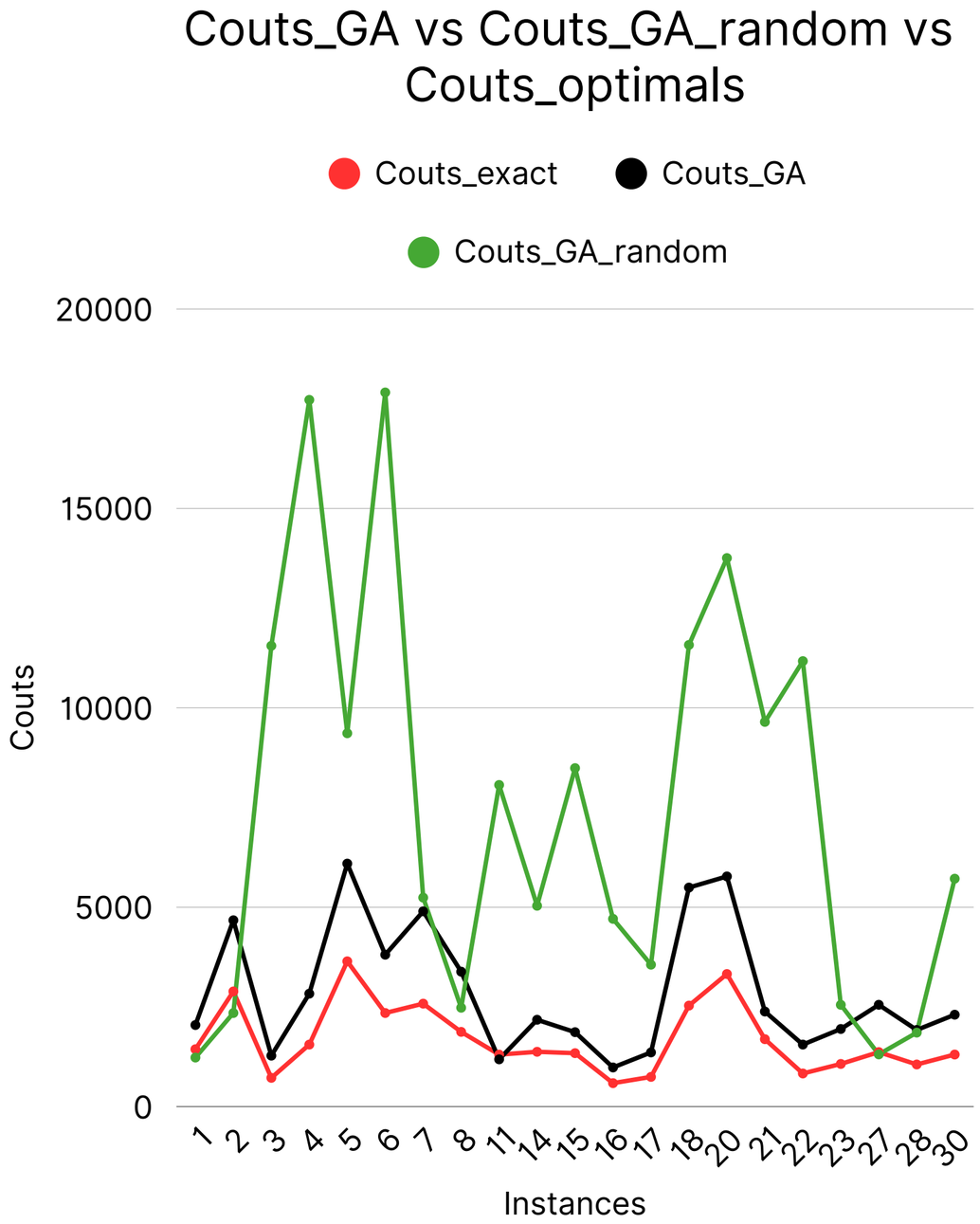
| Container_ID | Length | Destination |
|---|---|---|
| 1 | 3 | 1 |
| 2 | 4 | 4 |
| 3 | 1 | 5 |
| 4 | 4 | 5 |
| 5 | 5 | 2 |
| 6 | 4 | 3 |
| 7 | 4 | 3 |
| 8 | 5 | 3 |
| 9 | 1 | 2 |
| | 31 | |

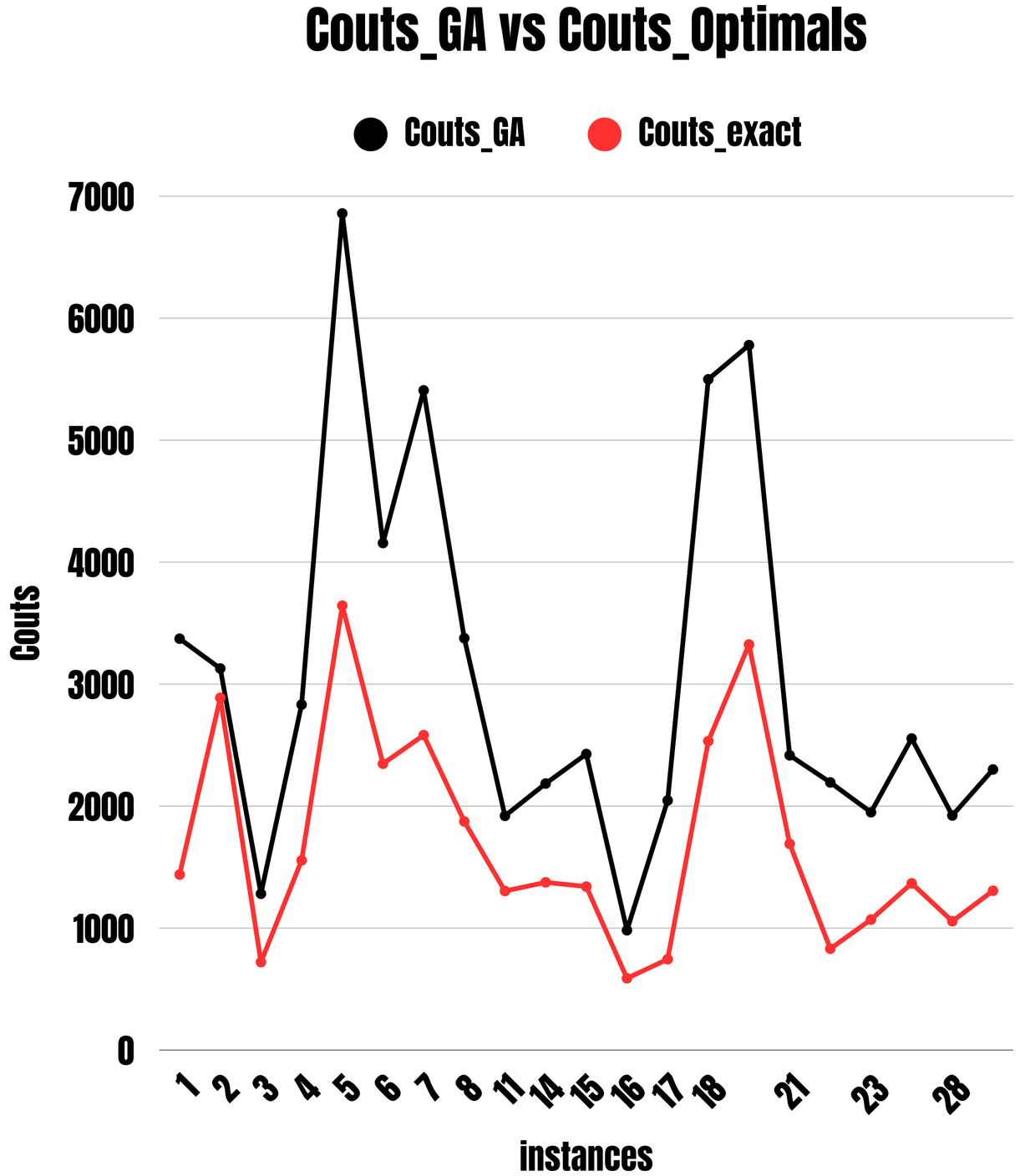
Trucks_max_capacity =6

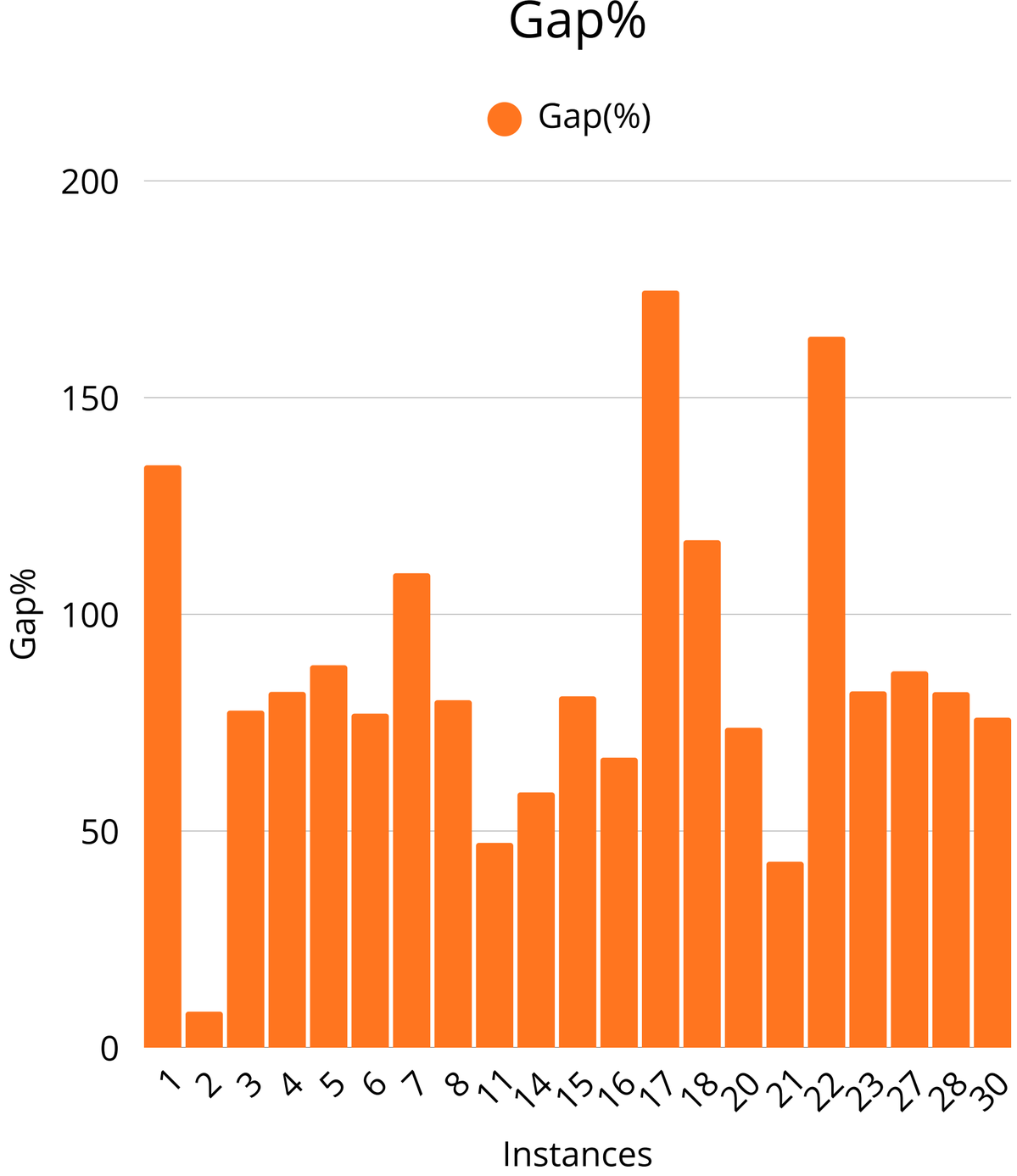
**Constraints**

- One truck can only load containers with same destination
- Containers (6, 7, 8) ->destination 3 ->3 trucks
- Containers (3, 4) ->destination 5 ->1 trucks
- Containers (5, 9) ->destination 2 ->1 trucks
- Containers (2) ->destination 4 ->1 trucks
- Containers (1) ->destination 1 ->1 trucks

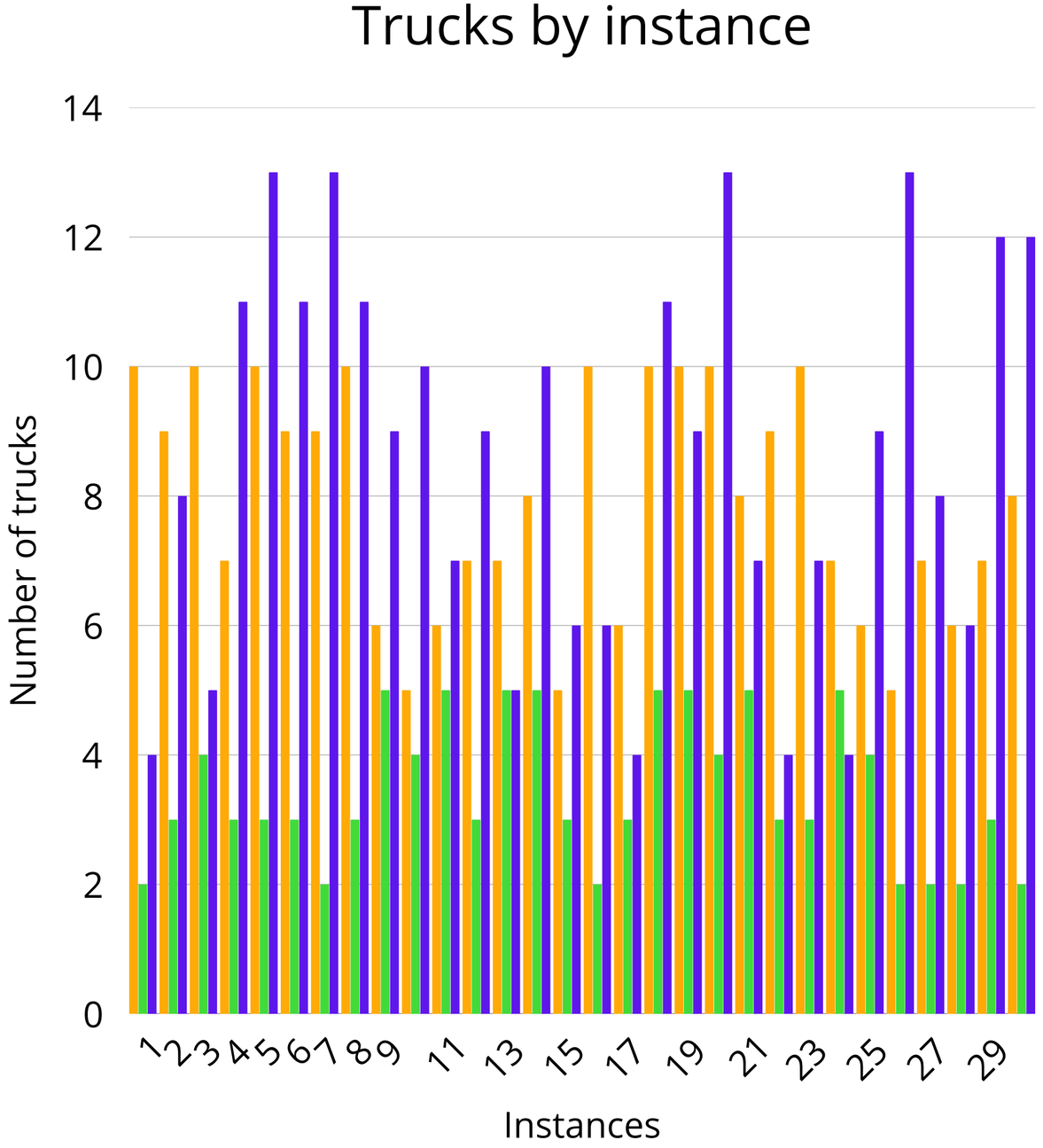
7 trucks> 6 trucks


Couts_GA vs Couts_GA_random vs Couts_optimals
Couts_exact
Couts_GA
Couts_GA_random
20000
15000
10000
5000
0
Couts
1 2 3 4 5 6 7 8 11 14 15 16 17 18 20 21 22 23 27 28 30
Instances


Couts_GA vs Couts_Optimals
Couts_GA
Couts_exact
7000
6000
5000
4000
3000
2000
1000
0
Couts
1 2 3 4 5 6 7 8 11 14 15 16 17 18 21 23 28
instances


Gap%
Gap(%)
200
150
100
50
0
Gap%
1
2
3
4
5
6
7
8
11
14
15
16
17
18
20
21
22
23
27
28
30
Instances


Time_GA vs Time_Exact
ExecutionTime(s)_GA
ExecutionTime(s)_Exact
1200
1000
800
600
400
200
0
temps (s)
1
2
3
4
5
6
7
8
11
14
15
16
17
18
21
23
28
instances


Trucks by instance
Number of trucks
0
2
4
6
8
10
12
14
1 2 3 4 5 6 7 8 9 11 13 15 17 19 21 23 25 27 29
Instances

# Conclusion & key  improvements to explore next

Initial initialization plays a big role on the quality of the GA solutions

Big instances->GA faster-> Lose precision

Big instances->Exact_methode found optimal-> bigger time execution

Tradeoff (Optimal solutions, and Running time)

> The results are still being stabilized, but the observed trends indicate that GA can provide a quick alternative, especially for large instances.

- **Move to Multi-Objective GA (NSGA-II)(Use NSGA-II to generate a Pareto front of best compromises)**
- **Parameter tuning : population size, crossover rate, mutation rate, elitism.**
- **Explore hybridized metaheuristics**

Thank you