Comparison with other algorithms :

**Hyperheuristic:**

* Time Efficiency: hyperheuristic time efficiency outperforms AG and B&B in terms of time efficiency, especially for larger and more complicated benchmarks.
* The number of bins used is usually moderate and consistent.

**Bound and Branch (B&B)**

* Time Efficiency: Efficient for small-scale problems, but it becomes impracticable due to very long calculation durations on larger benchmarks.
* The number of bins Efficient bin consumption is observed for smaller benchmarks, however it has limitations when attempting to solve more complex issues.

**Genetic Algorithm (AG)**

* Time Efficiency: This algorithm gives bad results for larger and more complex problems, resulting in increases in computation time. but, it is quite efficient for small to medium-sized benchmarks.
* Number of Bins Used: AG is effective in bin utilization for smaller problems, it loses efficiency as the problem size increases. For larger benchmarks, AG requires significantly more bins than other solutions

comparing between the previous methods we notice that :

* Hyperheuristic stands out as the best overall solution for a wide variety of problem sizes, notably for large and complex benchmarks. It is the most dependable and sensible option for common bin-packing difficulties since it combines excellent time efficiency with efficient bin utilization.
* AG lacks scalability but is extremely effective for smaller benchmarks. Its time and bin consumption efficiency drastically decreases with increasing problem size, making it less appropriate for more complicated and large-scale problems.
* For small-scale issues, B&B works incredibly well, providing ideal solutions with economical bin usage. Its weak scalability and lengthy processing times, however, make it impractical for bigger benchmarks, restricting its use to simpler cases.
* Simple heuristics solve small to medium large issues quickly and are straightforward to apply. But when it comes to scalability and bin usage, they usually perform worse than more advanced methods like AG and Hyper-heuristic.

In summary, the best results were given by the hyper-heuristic technique. Its ability to combine time efficiency with optimal bin usage makes it appropriate for a variety of problem sizes, including the most intricate ones. Although the Genetic Algorithm (AG) works well on smaller issues, it is less efficient and does not scale well on bigger benchmarks. For small, straightforward issues, Branch and Bound (B&B) works best; however, it becomes infeasible for larger ones. Heuristics are quick and easy to use, however they are not very scalable or efficient in using bins. Therefore, the hyperheuristic approach is the best option for maximum performance in a variety of settings.

Test of evaluation of the hyperheuristic :

Balancing optimality and complexity is crucial for developing effective and practical algorithms in this tests we set : complexity= 1 , optimality=1

| benchmark | population size | solution | execution time |
| --- | --- | --- | --- |
| 180 | 10 | 69 | 0.06798 |
| 180 | 100 | 73 | 0.35399 |
| 200 | 10 | 83 | 0.06304 |
| 100 | 100 | 87 | 0.36 |

**Population Size**:

* + Larger population sizes lead to worse solutions and require more execution time.where small size gives better result and execution time .

| benchmark | max number of iteration | solution | execution time |
| --- | --- | --- | --- |
| 200 | 10 | 82 | 0.0201 |
| 200 | 500 | 82 | 0.40665 |
| 200 | 1000 | 81 | 1.09977 |
| 200 | 9000 | 81 | 7.1983 |
| 200 | 15000 | 81 | 14.05955 |

**Taille de la population = 10**

**Taille du chromosome = 10**

**Number of Iterations**:

* + A small number of iterations gives a good result, however increasing the number of iterations beyond a certain point does not significantly improve the solution but increases execution time.

| benchmark | max number of iteration | solution | execution time |
| --- | --- | --- | --- |
| 1000 | 10 | 402 | 0.12663 |
| 1000 | 500 | 401 | 2.32654 |
| 1000 | 1000 | 401 | 5.08356 |
| 1000 | 9000 | 404 | 38.50908 |
| 1000 | 15000 | 403 | 65.37738 |

**Taille de la population = 10**

**Taille du chromosome = 20**

**Taille de chromosome**

* + varying the size of chromosome changes the solution performance in terms of time and number of bins,therefore an optimal chromosome size should balance solution quality and execution time.

| benchmark | HH solution | execution time | exact solution | execution time |
| --- | --- | --- | --- | --- |
| 50 | 25 | 0.00403 | 25 | 0.00099 |
| 100 | 47 | 0.0122 | 47 | 0.16797 |
| 180 | 68 | 0.018 | 68 | 0.0059 |
| 200 | 81 | 0.02 | / | very long execution time |

**Taille de population = 10**

**Taille de chromosomes = 20**

**Nombre d’itérations = 10**

Our tests showed that smaller population sizes and fewer iterations yield better solutions and faster execution times, while an optimal chromosome size enhances performance. The best combination for balancing these factors is a population size of 10, chromosome size of 20, and 10 iterations, achieving high-quality solutions efficiently