# Rental Bike Share Problem - Short Answers

Antonio G. L, Esteves

March 5, 2025

## 1 Task 2 - Solving with algorithms

The proposed algorithm was Variable Neighborhood Search (VNS). VNS is a metaheuristic optimization algorithm designed to escape local optima by systematically exploring different neighborhood structures. Unlike traditional local search methods that remain within a single neighborhood, VNS dynamically changes the neighborhood size during the search process, allowing it to explore a broader solution space. The algorithm operates by iteratively perturbing the current solution, applying a local search within the modified neighborhood, and accepting the new solution if it improves the objective function. If no improvement is found, the algorithm increases the neighborhood size, promoting diversification and enabling the search to escape suboptimal regions. This flexibility makes VNS highly effective for solving combinatorial and continuous optimization problems, particularly in scenarios where the search landscape is complex and multimodal.

VNS consists of several key components, including a shaking phase, local search, and neighborhood change mechanism. The shaking phase introduces controlled perturbations to the current solution, generating new starting points for the local search. The local search phase then refines the perturbed solution by exploring its immediate neighborhood using a predefined heuristic. If an improved solution is found, it replaces the current solution, and the search restarts from this new point. Otherwise, the algorithm transitions to a larger neighborhood, increasing the exploration radius. This systematic alteration between intensification and diversification strategies enables VNS to balance exploitation and exploration, making it a robust approach for various real-world applications, such as logistics, scheduling, and network design.

In addition to the proposed algorithm, two more algorithms were implemented in order to compare and test the performance and effectiveness of the current algorithm. In the following section, the complexities of the algorithms relative to time and auxiliary space, as well as their respective loop invariants, are presented.

## 1.1 What is the time complexity of your algorithm? What is the auxiliary space complexity of your algorithm?

The objective function of the proposed algorithm aims to maximize the total profit from relocating bicycles across different areas, considering that expected profits are not automatically accumulated. For each bicycle category $i$ and destination area $j$, the total profit is obtained by summing only the first $x_{i,j}$ values from the expected profit list $p_{i,j}$, where $x_{i,j}$ represents the number of relocated bicycles. Thus, the profit of each additional bicycle is computed independently, following the sequential order of available profit values. Additionally, the objective function imposes constraints to ensure that the number of relocated bicycles per category does not exceed the available surplus in the source area, penalizing infeasible solutions.

$$Fitness = profit - penalty \tag{1}$$

If a candidate solution violates constraints - exceeding the surplus for a category or the truck capacity - the penalty (multiplied by a high factor, e.g., 1000) can be very large. This subtraction can result in a negative fitness even if the profit is positive, especially when penalties dominate. In essence, negative fitness values indicate that the solution is not feasible or is far from feasible. Adjusting the penalty factors or ensuring solutions respect constraints can help improve the fitness values.

### 1.1.1 VNS

The VNS algorithm presented is structured in such a way that, at each iteration of the outer loop, a series of "shaking" operations and local search procedures are performed to improve the current solution. At the beginning of each iteration of the outer loop, the best solution found up to that point is stored along with its objective function value. The stored solution must be at least as good as any other solution explored throughout the algorithm. Thus, any improvement obtained through neighborhood exploration and subsequent local search is immediately incorporated, resetting the iteration counter for non-improving solutions.

The presented algorithm has a time complexity dominated by the maximum number of iterations $N$, the number of neighborhoods explored $K$, and the number of local search iterations $M$. In each iteration, a new solution is generated and evaluated, resulting in an execution time of:

$$O(N \times K \times M \times n) \tag{2}$$

where $n$ represents the size of the solution. The evaluation of the objective function and the generation of new solutions significantly contribute to the computational cost, making the algorithm's performance sensitive to the efficiency of the perturbation and evaluation functions.

The auxiliary space complexity of the algorithm is dominated by storing the current solution, the best solution, and the fitness history over the iterations. The required memory is:

$$O(n + N) \tag{3}$$

where $n$ corresponds to the size of the solution and $N$ to the maximum number of iterations, due to the storage of the sequence of best fitness values found. Thus, the main memory-limiting factor is the fitness history, which can be reduced to optimize space consumption.

### 1.1.2 Genetic Algorithm

The proposed algorithm ensures that, in each iteration of the main loop, the population maintains a constant size of $P$ individuals, and the best solution found so far is correctly stored. Additionally, the fitness history records the evolution of the best fitness value in each generation, ensuring the traceability of the algorithm's performance. The new population is always generated from the previous one through selection, crossover, and mutation operators, preserving the coherence of the evolutionary process. These properties remain valid throughout all iterations, guaranteeing the algorithm's stability until the stopping criterion is met.

The presented genetic algorithm has a time complexity dominated by the number of generations ($G$) and the population size ($P$). In each generation, the fitness evaluation of individuals occurs in $O(P \times f(n))$, where $f(n)$ represents the time required to evaluate the objective function. The tournament selection, crossover, and mutation processes also impact performance, but each of these operations runs in $O(1)$ time per individual. Since each generation involves $O(P)$ evaluations and genetic operations, the total time complexity of the algorithm is $O(G \times P \times f(n))$, being strongly influenced by the cost of the evaluation function.

The auxiliary space complexity of the algorithm is determined by storing the population, candidate solutions, and fitness history. The memory consumption is dominated by the population, which requires $O(P \times n)$ to store $P$ individuals, each with $n$ genes. Additionally, the fitness history stores $O(G)$ values throughout the generations. Thus, the space complexity is $O(P \times n + G)$, primarily limited by the population size and the representation of the solution.

### 1.1.3 Tabu Search

The proposed algorithm ensures that, in each iteration, the current solution is always the best viable candidate found within the explored neighborhood while respecting the constraints of the tabu list. Additionally, the global best solution is updated only if a candidate with a superior fitness value is identified. The tabu list maintains a limited history of recently visited solutions, ensuring search diversification. These properties are preserved throughout all iterations, guaranteeing that the algorithm continues exploring new regions of the search space until the stopping criterion is met.

The proposed algorithm's time complexity is dominated by the maximum number of iterations ($I$) and the neighborhood size ($N$). In each iteration, $N$ candidates are generated and evaluated, where each evaluation has a cost of $O(f(n))$, with $f(n)$ representing the time required to evaluate the objective function. Furthermore, searching for the best candidate requires a linear scan of the neighborhood, resulting in a cost of $O(N)$. Thus, the overall time complexity of the algorithm is $O(I \times N \times f(n))$, directly influenced by the evaluation function's cost and the size of the explored neighborhood.

### 1.2 The algorithm you proposed in letter a) will always find an optimal solution, regardless of the input? How?

No, (unfortunately )this VNS algorithm does not guarantee finding the optimal solution for all possible inputs. However, it is designed to search for high-quality solutions and has a good chance of finding near-optimal solutions, depending on the problem structure and the number of iterations allowed. Despite that, given unlimited time, VNS would theoretically be able to reach the optimal solution by exhaustively exploring all neighborhoods.

### 1.3 By analyzing the input data, which preprocessing task can be made to this instance in order to make it faster or lighter to be solved?

By analyzing the input data several preprocessing steps can be taken to make the problem faster and lighter to solve:

- Remove categories with zero surplus.
- Ignore Areas with No Expected Profits.
- Check for sparsity and Order profits.
- Precompute Cumulative Profits for Faster Lookup.
- Categories with higher expected profits per unit space should be prioritized in allocation.

## 2 Task 3 - Going further on task 2 (problem without the capacity constraint). Here it is not necessary to code, only explain your answers

### 2.1 How can parallel processing be applied for reducing the run time of solving this problem?

Parallel systems offer an increase in the performance of slow programs, natural solutions for intrinsically parallel programs and a possible modularity of programs. In general, the paradigm presents some difficulties to be considered, such as programming difficulty, the need for load balancing, communication and synchronization between processes. Thus, to this problem specifically, we can apply parallelism in the execution of the code by parallelizing the sum of the expected profits, since calculating the total profit for each reallocated bicycle is independent for each category and area (it is also possible to use multiprocessing or vectorization techniques to speed up decisions).

### 2.2 If it is assured that the number of bicycles to be redistributed will always be at least 90% of the expected demand for all areas, can you adapt your algorithm to use this property for running faster?

Normally we would need to test all possible quantities of bikes reallocated to each area, since we know that at least 90% of the demand will be met, we can start by allocating this amount directly and reduce the number of possible combinations. That is, if we explore all possible quantities beforehand, we now know that at least 90% of the demand will be allocated and we adjust only if there is capacity or need.

### 2.3 Now the company RentalBike wants to avoid relocating to an area that is not prioritized, unless other regions are reasonably served. What must be changed or included in the formulation you proposed, at Task 1 a), to assure that the area number 1 will only receive a bicycle if at least 85% of the surplus of each category is already allocated to other areas?

In this case, we want area 1 to receive bicycles only after 85% of the surplus in each category has been allocated to other areas. To do this, we must add the following constraint:

$$\sum_{j \neq 1} x_{ij} \geq 0.85 \times S_i, \quad \forall i \tag{4}$$

This constraint ensures that for each category i, at least 85% of the total available is distributed to other areas before area 1 receives any bicycles.

### 2.4 Now RentalBike wants to balance the profits among areas. What must be changed in your formulation to maximize the expected profits of the region with lowest expected profits, instead of simply maximizing the summation of profits?

In this case, we want to ensure that the area with the lowest profit has its profit maximized. Therefore, we can introduce an auxiliary variable L that represents the lowest expected profit among all areas.

$$L \leq \sum_i P_{ij} \times x_{ij}, \quad \forall j \tag{5}$$

In other words, $L$ will always be less than or equal to the expected profit of any area j. The objective now would be to maximize $L$, ensuring that the profit of the weakest area is as high as possible.

## 3 Task 4 - Going further on business. Here it is not necessary to code, only explain your answers

### 3.1 Which other analysis, optimizations or policies do you suggest to RentalBike that could bring business impact?

- Analysis of variation by category, as some categories appear more advantageous depending on the area.
- Instead of redistributing bicycles purely based on expected profit, consider demand fluctuations due to:
  - Commuting hours (morning/evening rush).
  - Weather conditions (bikes are less used on rainy days).
  - Special events (concerts, sports events).
- Addition of demand forecasting with a forecasting model.
- Study of forecasts for a dynamic pricing system (like Uber).

- Incentives for users who return bikes in specific locations (areas with lower bike availability).
- Expanding Business Through B2B Partnerships.