Algorithmic Methods for Mathematical Models
Course Project

Marco Bartoli, Ilan Vezmarovic

UPC Universitat Politècnica de Catalunya

May 26, 2024

Formal Problem Definition

- n: number of products
- x: height of the suitcase in millimeters
- y: width of the suitcase in millimeters
- c: limit to the total weight of the suitcase in grams
- p_i: price of the i-th product in euros
- w_i: weight of the i-th product in grams
- s_i: side length of the i-th product's (square) box in millimeters

Decision Variables

- Chosen_i: binary variable that is 1 if object i is chosen, and 0 otherwise.
- PointsX_i: the x-coordinate of the bottom-left corner of object i.
- PointsY_i: the y-coordinate of the bottom-left corner of object i.
- $\text{Overlap}_{i,j,d}$: binary variable indicating if objects i and j do not overlap in direction d, where $d \in \{1, 2, 3, 4\}$.

Objective Function

Maximize the total price of the chosen objects:

$$\text{maximize} \sum_{i=1}^{n} p_i \cdot \text{Chosen}_i$$

Max Weight Constraint

Ensure the total weight of the chosen objects does not exceed the suitcase's capacity:

$$\sum_{i=1}^{n} w_i \cdot \text{Chosen}_i \leq c$$

Coordinate Bounds Constraints

Ensure each object lies entirely within the suitcase's boundaries:

$$\forall i \in \{1, \ldots, n\}, \quad \mathrm{PointsX}_i \geq 1$$

$$\forall i \in \{1, \ldots, n\}, \quad \mathrm{PointsY}_i \geq 1$$

$$\forall i \in \{1, \ldots, n\}, \quad \mathrm{PointsX}_i + s_i - 1 \leq x$$

$$\forall i \in \{1, \ldots, n\}, \quad \mathrm{PointsY}_i + s_i - 1 \leq y$$

Non-Overlapping Constraints

Left/Right/Up/Down non-overlapping:

$$\forall i, j \in \{1, \dots, n\}, i \neq j, \quad \text{PointsX}_i - \text{PointsX}_j + s_i \leq -M \cdot (\text{Chosen}_i + \text{Chosen}_j + \text{Overlap}_{i,j,1} - 3)$$

$$\forall i, j \in \{1, \dots, n\}, i \neq j, \quad \text{PointsX}_i - \text{PointsX}_j + s_i \leq -M \cdot (\text{Chosen}_i + \text{Chosen}_j + \text{Overlap}_{i,j,2} - 3)$$

$$\forall i, j \in \{1, \dots, n\}, i \neq j, \quad \text{PointsX}_i - \text{PointsX}_j + s_i \leq -M \cdot (\text{Chosen}_i + \text{Chosen}_j + \text{Overlap}_{i,j,3} - 3)$$

$$\forall i, j \in \{1, \dots, n\}, i \neq j, \quad \text{PointsX}_i - \text{PointsX}_j + s_i \leq -M \cdot (\text{Chosen}_i + \text{Chosen}_j + \text{Overlap}_{i,j,4} - 3)$$

Non-Overlapping Constraints

At least one of the non-overlapping conditions is satisfied:

$$\forall i, j \in \{1, \ldots, n\}, i \neq j, \quad \sum_{d=1}^{4} \text{Overlap}_{i,j,d} \geq 1$$

Greedy heuristic

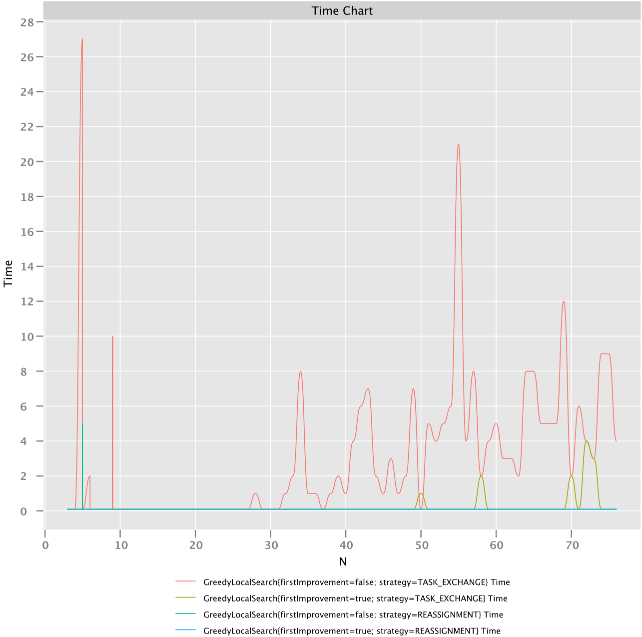
Input: A problem instance with n products, suitcase dimensions (x, y), and weight capacity c.
Output: A feasible solution with selected products that maximize the total price.

Initialize solution as empty
Sort products by $(price/side/weight)$ in descending order
for each product in sorted products **do**
    if product can fit in the suitcase and does not exceed weight limit **then**
        Add product to solution
        Update suitcase dimensions and weight capacity
    end if
end for
return solution

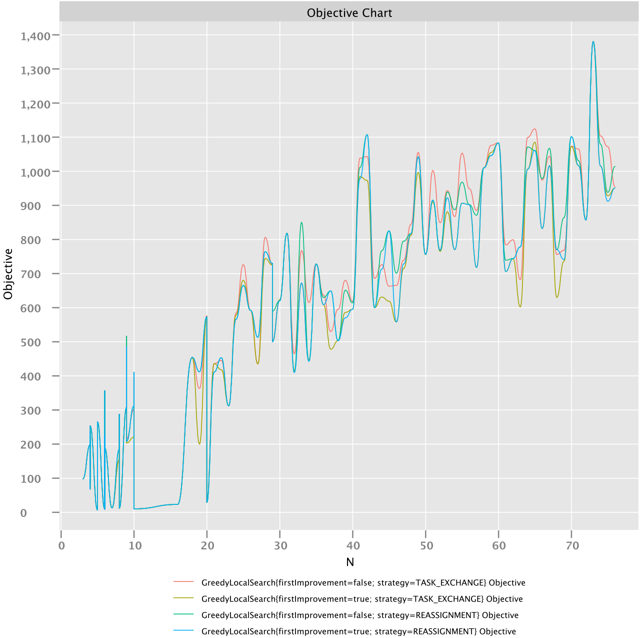
Local search

```
Input: An initial solution, mode and strategy.
Output: An improved solution
bestSolution = initialSolution
improved = true
while improved do
    improved = false
    for each product in bestSolution do
        if strategy == EXCHANGE then
            newSolution = exchangeProduct(bestSolution, product)
        else if strategy == REASSIGNMENT then
            newSolution = reassignProduct(bestSolution, product)
        end if
        if newSolution is better than bestSolution then
            bestSolution = newSolution
            improved = true
            if mode == FIRST_IMPROVEMENT then
                return bestSolution
            end if
        end if
    end for
end while
return bestSolution
```

Local Search Time chart



Local Search Objective chart



GRASP

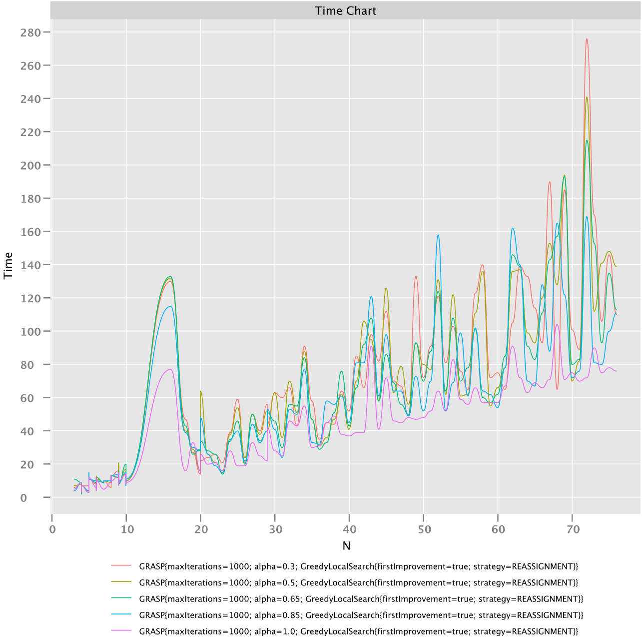
Input: A problem instance, maxIterations, alpha (for RCL threshold)
Output: The best solution found

```
bestSolution = null
for iteration = 1 to maxIterations do
    greedySolution = constructGreedyRandomizedSolution(problem, alpha)
    localOptimalSolution = LOCAL_SEARCH(greedySolution, FIRST_IMPROVEMENT)
    if bestSolution is null or localOptimalSolution is better than bestSolution then
        bestSolution = localOptimalSolution
    end if
end for
return bestSolution
```

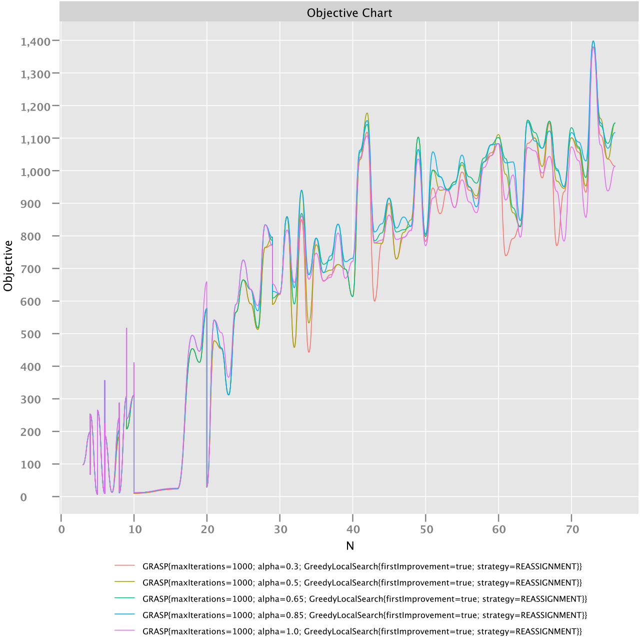
GRASP (2)

Input: A problem instance, alpha (for RCL threshold)
Output: A feasible solution
Initialize solution as empty
Sort products by $(price/side/weight)$ in descending order
while there are remaining products **do**
    $qMax$ = maximum Q value in remaining products
    $qMin$ = minimum Q value in remaining products
    threshold $= qMax - alpha \times (qMax - qMin)$
    RCL $= \{$products with Q value $>=$ threshold$\}$
    selectedProduct = randomly select a product from RCL
    if selectedProduct fits in the suitcase and does not exceed weight limit **then**
        Add selectedProduct to solution
        Update suitcase dimensions and weight capacity
    end if
end while
return solution

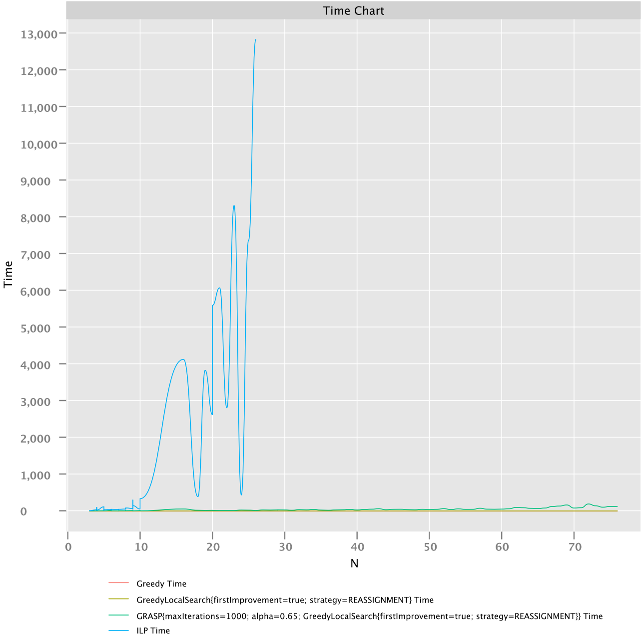
GRASP Time chart



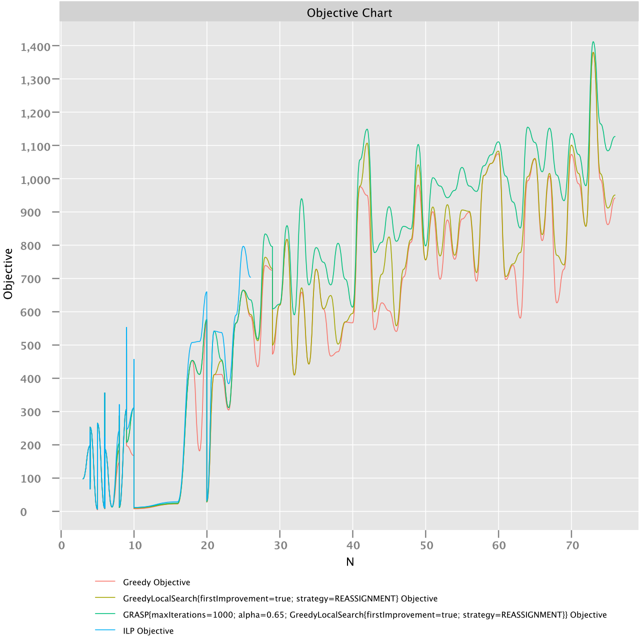
GRASP Objective chart



All Time chart



All Objective chart



Improvements

- Parallelize algorithm
- Better strategies
- Implement the state of the art for heuristic