Algorithmic Methods for Mathematical Models Course Project

Mathematics of Baking

Tim Wichelmann Jakob Eberhardt

December 19th

Instance Input Data

- Number of items (`n`)
- Time periods (`t`)
- Profits (`profit`)
- Lengths (`length`)
- Minimum deliveries (`min_deliver`)
- Maximum deliveries (`max_deliver`)
- Surfaces (`surface`)
- Surface capacity (`surface_capacity`)

Decision Variables and Objective Function

- y is a matrix of size $n \times t$ consisting of binary variables which indicate if the order i will be baked in time slot j.
- x_i is a binary variable indicating whether order i has the right amount of time slots assigned to it.
- $start_i$ denotes the time slot in which the baking process of order i is started.
- end_i denotes the time slot in which the baking process of order i will be finished.

$$\max \sum_{i=1}^{n} profit_i \times x_i$$

Goal: The objective is to maximize the total profit, calculated as the sum of profits of each order that is successfully scheduled while respecting the constraints.

Surface Constraint

Problem: The naive schedule would exceed our oven capacity
Constraint: In every time slot, the space capacity is respected.

$$\sum_{i=1}^{n} surface_i \, y_{ij} \leq surface\_capacity, \qquad (1 \leq j \leq t) \tag{1}$$

Continuity & Length Constraint

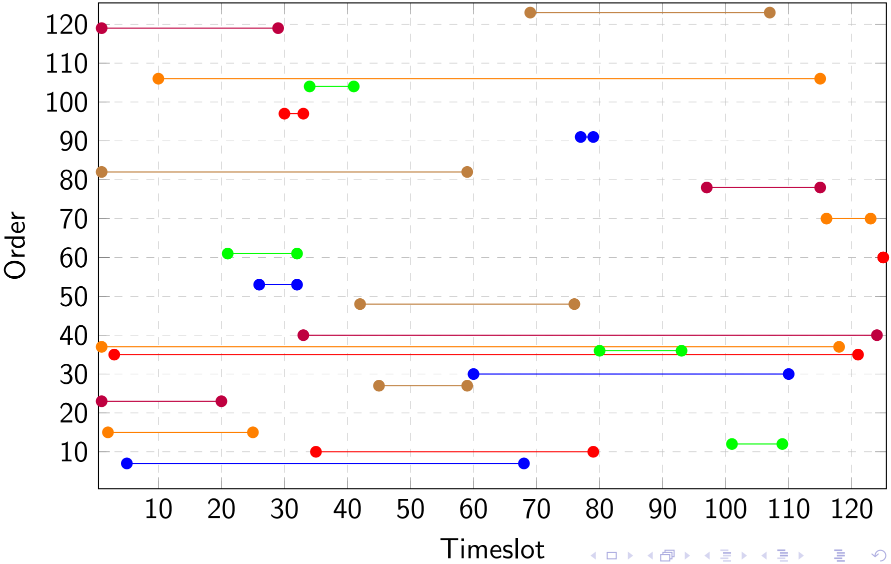
Problem: Once started, an order has to be continuously processed for the needed time lots associated with it.
Constraint: If an order i is part of the schedule, it is assigned the $length_i - 1$ contiguous time slots from $start_i$ to end_i:

$$y_{ij} = (j \geq start_i) \wedge (j \leq end_i), \qquad (1 \leq i \leq n, 1 \leq j \leq t). \qquad (2)$$

Can also be done without logic operators (see report).
Optimization: Check only feasible slots for order i.

Lets consider this optimal solution

Objective function value: 164.



Greedy cost function

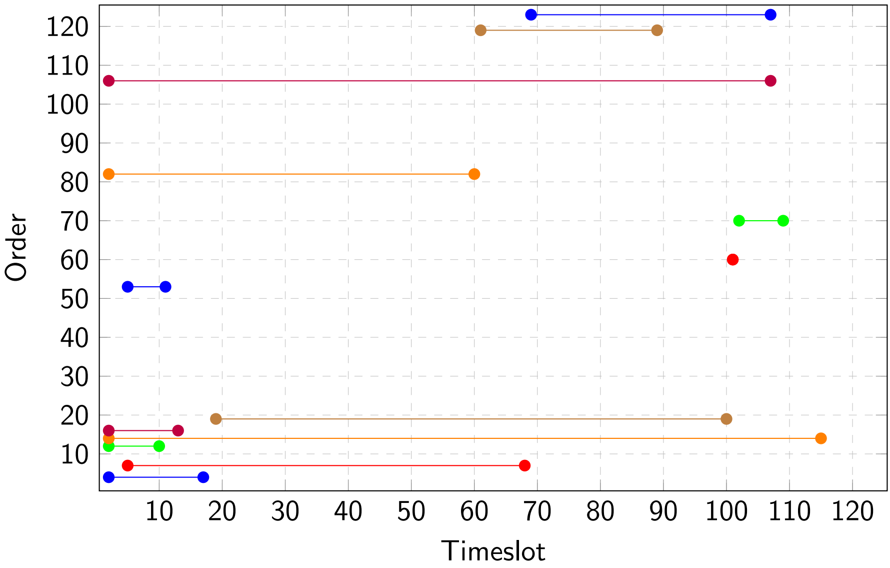
Greedy cost function is split into two parts:

- For an order i: Only consider its profit.
- For a possible starting time slot j (for a fixed order i): The average percentage of space used during the baking

$$\frac{\sum_{l=j}^{j+length_i}(\sum_{k=1}^{n} y_{kj} surface_k / surface\_capacity)}{length_i}$$

- Objective function value: 97
- **Optimality gap: 41%**

Greedy only considering profit



Greedy cost function

Improvement for greedy

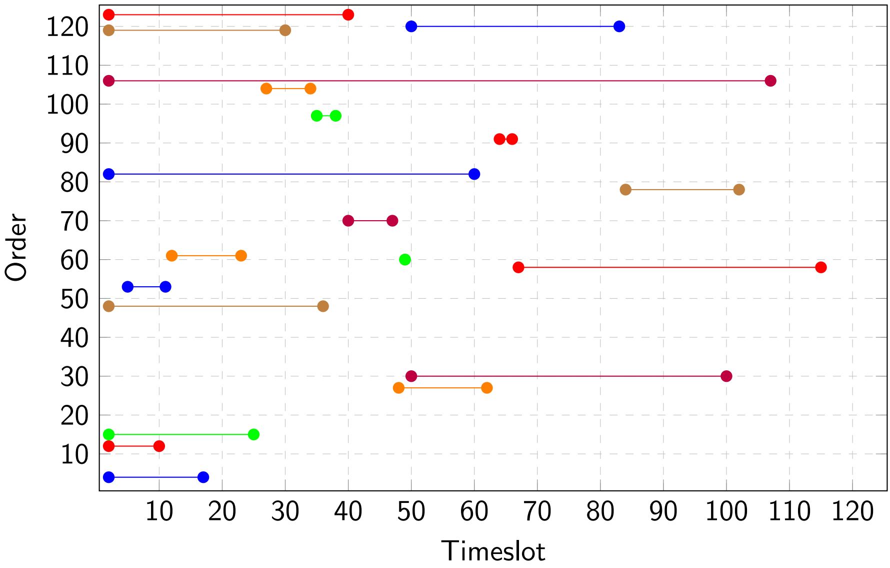
For an order i:

- Also consider other attributes:

$$\frac{profit_i(max\_deliver_i - min\_deliver_i)}{(length_i \cdot surface_i)}$$

- Objective function value: 126
- **Optimality gap: 23%**

Scoring the orders



Local search criteria

Local Search

- Remove the order with the highest surface from the current schedule.
- Try to fit in the previously unused orders (in order of descending profit).

- 45 seconds of local search
- Objective value: 127
- **Optimality gap: 22.5%**

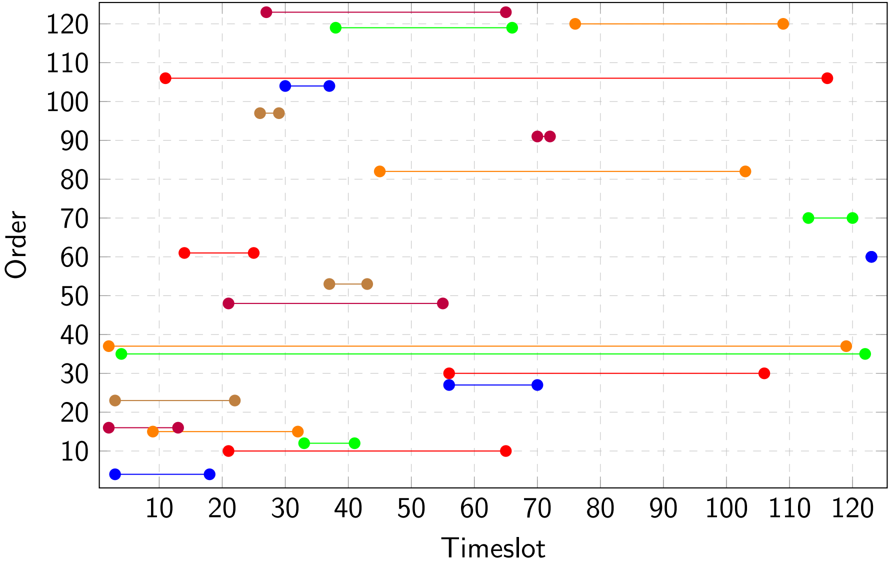
Greedy cost function

GRASP

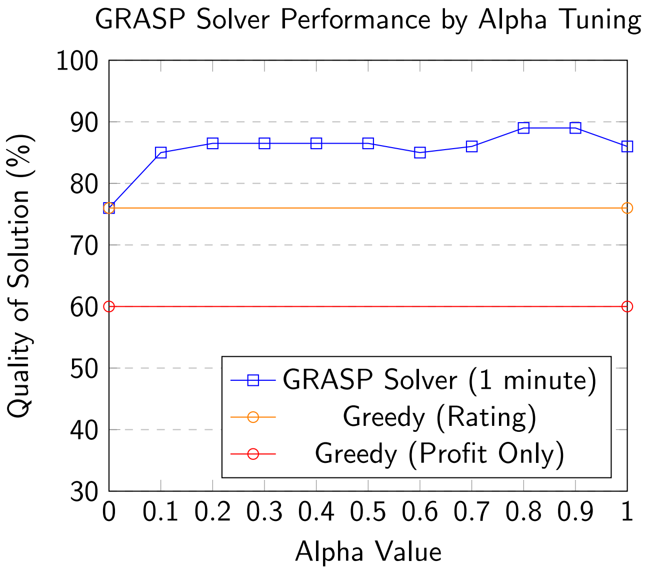
- Best results: select orders the same way as in greedy, according to their greedy cost function.
- Use an RCL with parameter α for selecting the time slot.

- 60 seconds, $\alpha = 0.8$
- Objective value: 146
- **Optimality gap: 11%**

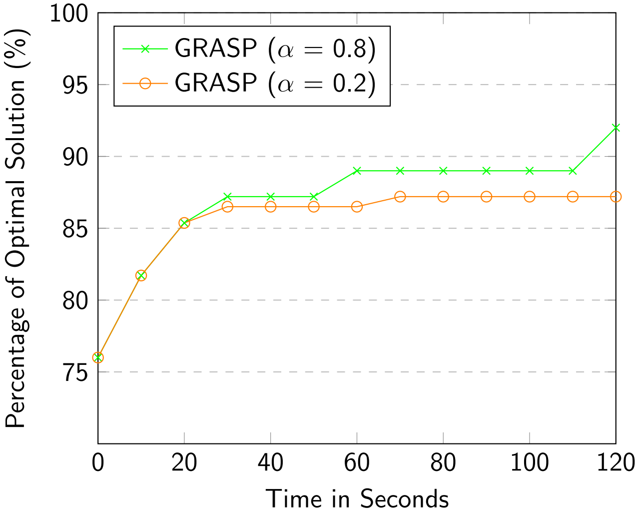
GRASP after 1 minute and 5 improvements



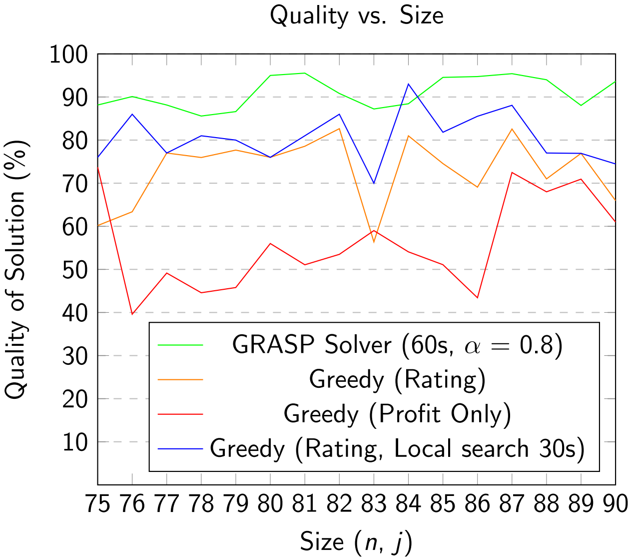
Tuning the α-parameter



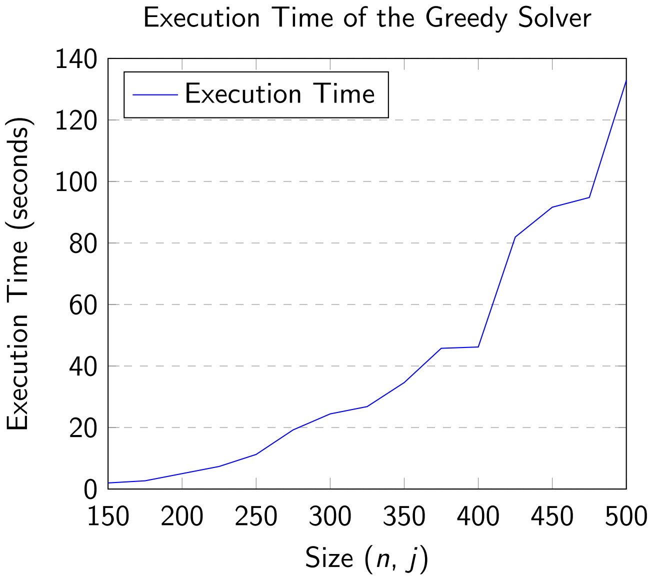
Time and α-parameter



Problem size vs. solution quality

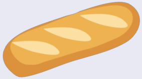


Greedy and very big instances



Conclusion and Future Prospects

Results



- Developed the model
- Obtained optimal solutions using CPLEX
- Implemented and benchmarked different greedy cost functions
- Improved the greedy solutions using local search
- Applied the GRASP meta-heuristic
- Tuned α towards a consistent single-digit optimality gap in 60 seconds

Additional Considerations

- Greedy: Try different combinations of order/time slot greedy cost functions.
- GRASP: Apply the RCL to the list of sorted orders to explore more diverse solutions.

Backup Slides

Continuity & Length Constraint

Problem: Once started, an order has to be continuously processed for the needed time lots associated with it.

Constraint:

If an order i is part of the schedule, it is assigned the $length_i - 1$ contiguous time slots from $start_i$ to end_i:

$$j \geq start_i - (t+1) * (1 - geq\_start_{ij}) \tag{3}$$

$$start_i \geq j - (t+1) * geq\_start_{ij} \tag{4}$$

$$end_i \geq j - (t+1) * (1 - leq\_end_{ij}) \tag{5}$$

$$j \geq end_i - (t+1) * leq\_end_{ij} \tag{6}$$

$$0 \leq geq\_start_{ij} + leq\_end_{ij} - 2 * y_{ij}, \qquad (1 \leq i \leq n, min\_start_i \leq j \leq max \tag{7}$$

CPLEX execution time