GRASP Method for Vehicle Routing with Delivery Place Selection

Petar Afric, Adrian Satja Kurdija, Lucija Sikic, Marin Silic, Goran Delac, Klemo Vladimir, and Sinisa Srbljic

Consumer Computing Lab
Faculty of Electrical Engineering and Computing
University of Zagreb
Croatia

2019 International Conference on AI and Mobile Services
June 25 - June 30 2019, San Diego, USA

Introduction

Introduction

- Problem
  - Package delivery with delivery site selection
- Motivation
  - Increasing need for optimized package delivery
  - Maximize profit
  - Reduce our impact on the environment
- Inspiration
  - Schittekat et al. paper on school bus routing (SBR) with bus stop selection.
  - Multiple delivery sites produce shorter routes - introduce it to package delivery.
  - Large running time of the proposed algorithm.
  - Very simple benchmark model.

Problem Definition

Problem

- Defining vehicle routes with delivery place selection
- Single production factory
- N delivery stations
- M customers
- Constraints:
  - Each customer will visit only one delivery station within walking range
  - Each customer will pick up only one delivery package
  - All delivery trucks have the same capacity given in number of packages
  - Any delivery station can be visited by only one delivery truck
- Goal
  - Minimize the total travelled length

Mathematical description

Parameter	Description
N	number of delivery stations
M	number of customers
T	number of delivery trucks
K_k	capacity of delivery truck k specified by number of packages it can carry
c_{ij}	cost of going from station i to station j
cp_i	cost of going from station i to the production factory
pn_l	number of packages picked up by customer l
s_{il}	1 if the station i is within range of customer l, otherwise 0
x_{ijk}	1 if delivery truck k travels between stations i and j, otherwise 0
xp_{ik}	1 if delivery truck k travels between stations i and the production factory, otherwise 0
y_{ik}	1 if the delivery truck k goes to station i, otherwise 0
z_{il}	1 if the customer l goes to station i, otherwise 0

Mathematical description

$$\sum_{i=1}^{N}\sum_{j=1}^{N} c_{ij} \sum_{k=1}^{T} x_{ijk} + \sum_{i=1}^{N} cp_i \sum_{k=1}^{T} xp_{ik} \tag{1}$$

$$\sum_{i=1}^{N} z_{il} s_{il} = 1 \quad \forall l \in \{1, ..., M\} \tag{2}$$

$$pn_l = 1 \quad \forall l \in \{1, ..., M\} \tag{3}$$

$$K_i = K_j \quad \forall i, j \in \{1, ..., T\} \tag{4}$$

$$\sum_{k=1}^{T} y_{ik} = 1 \quad \forall i \in \{1, ..., N\} \tag{5}$$

Proposed Algorithm

Overview

- Fast greedy randomized adaptive search procedure (GRASP)
- Two parts:
  - Assigning customers to delivery stations - a greedy heuristic
  - Defining routes for visiting delivery stations - a specialized local search
- Repeat the optimization several times

Assigning customers to delivery stations idea

- Reduce the number of delivery stations
- Preferring delivery stations which are closer to the production factory
- Active station - that which has a customer assigned

Assigning customers to delivery stations overview

- Calculate the fitness of each station
- Sort the stations in descending order by fitness.
- If the fitness difference is less then *fitness_d_min* prefere the closer station
- Iterate over the sorted stations and assign customers to them

Initial station fitness calculation procedure

- For each station initialize to zero.
- Generate constant C uniformly from $[0, 10]$
- To each fitness add C divided by the station-factory distance
- For each customer calculate the reciprocal of the number of reachable stations
- Add that to the fitness of every reachable station

Assigning customers to delivery stations procedure

- If the station can be reached by a number of customers less or equal to the capacity of the truck, then all the customers are assigned to that station.
- If more customers can reach the station:
  - They are sorted by the number of stations they can reach in ascending order
  - If two customers can reach the same number of stations, precedence is given to the customer which is farther from the production factory.
  - Customers are then assigned to the current station in the sorted order until the capacity on the truck is filled.

Defining routes to delivery stations

- Define truck routes for active stations
- Make them as short as possible while respecting constraints
- Problem division:
  - Which stations are in the same route
  - The best possible order of visiting the stations
- Algorithm division:
  - Preprocessing
  - Initial solution generation
  - Solution optimization

Defining routes to delivery stations preprocessing

- Iterate over all stations
- Defining routes for stations which cannot be combined with any other station due to the constraints
- These stations and routes are no longer taken into consideration

Initial solution generation

- Greedy heuristic

1. Select a station which is not assigned to any route
2. Creates a route for it
3. Iterate over the rest of the unassigned stations
4. Calculate which of the *satisfiable* stations is closest to the route
5. *Satisfiable* station - the station-route distance is smaller than the station-factory distance
6. The closest satisfiable station is added to the current route is ti exists
7. Otherwise the current route creation ends
8. The previous steps are repeated as long as there are active unassigned stations

Solution optimization

- Local search
- An incomplete neighborhood consists of:
  - Switching a station between routes
  - Selecting two stations from different routes and swapping them
  - Joining two routes
  - Randomly breaking a route into two
- With small probability another modification occurs after each modification
- Routes are re-optimized using a TSP solver in each iteration
- The objective function is the sum of lengths of all routes.
- The stopping condition
  - Maximum number of iterations
  - Maximum number of stagnant iterations

TSP solver

- Greedy heuristic
- Given three or less nodes return
- Otherwise, three random nodes are declared the current optimal route.
- Then:
    - Select a random node,
    - Iterate over each edge of the current optimal route and calculate the distance change
    - Remove the edge which produced the minimal change in distance.
- Repeated until all nodes are in the route.

Results

Dataset

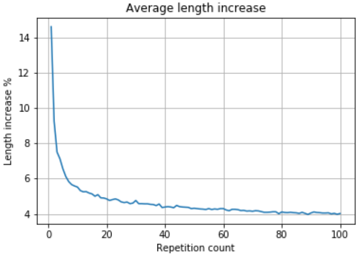
- 112 problem instances
- number of customers ranging from 25 to 800
- number of stations ranging from 5 to 80
- the maximum allowed walking distance ranging from 5 to 40

Evaluation

- For every value of the repetition parameter instances are solved 100 times
- Calculate the average solution route length
- Calculate the average calculation time
- Compare the results to those presented in [1]

Solution length results

- At 100 repetitions our algorithm produces, on average, 4% longer routes
- 23% longer routes by baseline in [1]
- At 100 repetitions as the maximum walking distance varies from 5 to 40, the increase in the route length varies from -0.54% to 4.5%
- From 1.4% to 63.4% by baseline in [1]
- At 100 repetitions as the amount of stations varies from 5 to 80, the increase in the route length for our solutions varies from 0% to 8.31%
- At 100 repetitions as the amount of customers varies from 25 to 800, the increase in the route length for our solutions varies from 0% to 6.98%



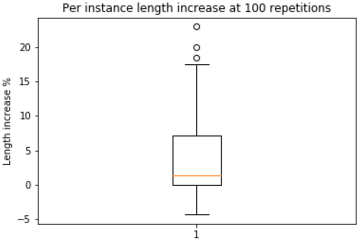
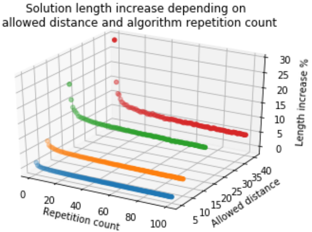


Figure: Average solution route length increase



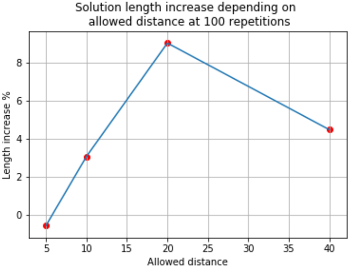


Figure: Solution quality in relation to walking distance

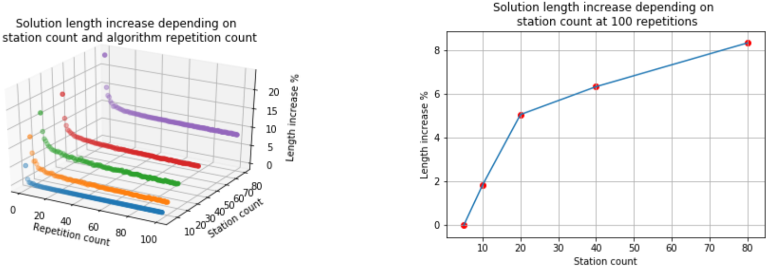


Figure: Solution quality in relation to station count

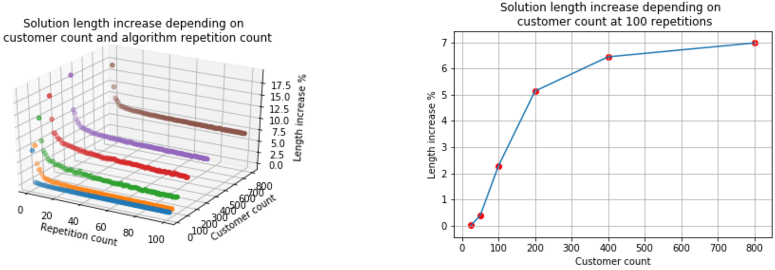
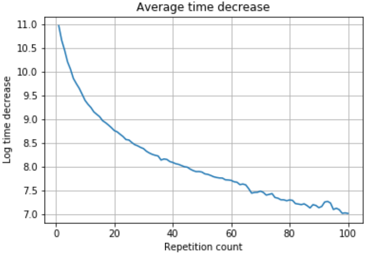


Figure: Solution quality in relation to customer count

Solution time performance

- The time performance results depict the logarithm of time decrease
- At 100 repetitions our algorithm, on average takes $e^{7.02} \approx 1100\times$ less time
- At 100 repetitions as the maximum walking distance varies from 5 to 40, the time decrease for our algorithm varies from $e^{7.07} \approx 1171.13\times$ to $e^{6.82} \approx 913.75\times$
- At 100 repetitions as the number of stations varies from 5 to 80, the time decrease for our algorithm varies from $e^{2.01} \approx 7.47\times$ to $e^{8.53} \approx 5036\times$
- At 100 repetitions as the number of customers varies from 25 to 800, the time decrease for our algorithm varies from $e^{5.11} \approx 165\times$ to $e^{7.8} \approx 2436\times$



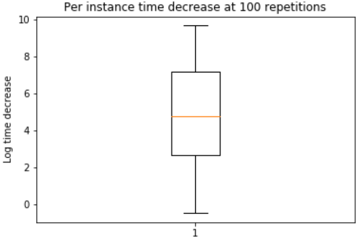
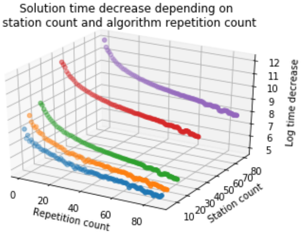


Figure: Average time decrease



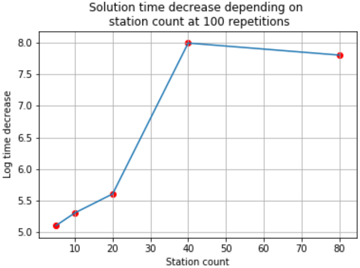
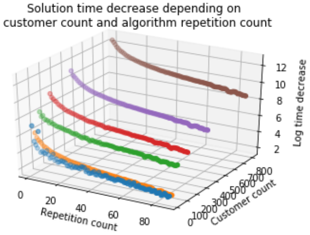


Figure: Solution time decrease in relation to station count



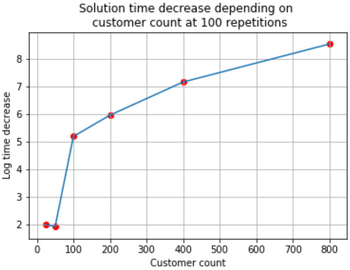


Figure: Solution time decrease in relation to customer count

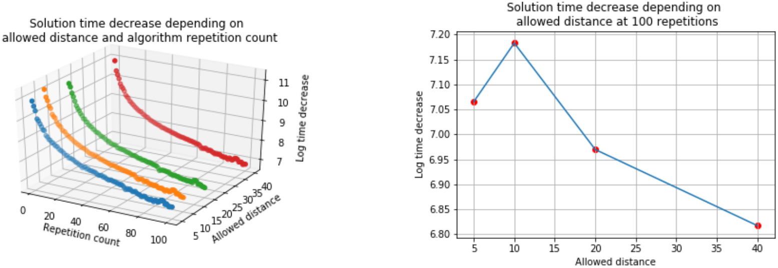


Figure: Solution time decrease in relation to maximum walking distance

References I

[1] *Schittekat, P., Kinable, J., Srensen, K., Sevaux, M., Spieksma, F., Springael, J.*, A metaheuristic for the school bus routing problem with bus stop selection *European Journal of Operational Research*, **Vol 229**, 2013.