Insert here your thesis' task.

Czech Technical University in Prague
Faculty of Information Technology
Department of Theoretical Computer Science

Bachelor's thesis

Vehicle routing problem, its variants and solving methods

Michal Polívka

Supervisor: RNDr. Tomáš Valla, Ph.D.

7th April 2015

Acknowledgements

THANKS

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as school work under the provisions of Article 60(1) of the Act.

In Prague on 7th April 2015 .....

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Polívka, Michal. *Vehicle routing problem, its variants and solving methods.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2015.

Abstrakt

V několika větách shrňte obsah a přínos této práce v českém jazyce.

Klíčová slova Replace with comma-separated list of keywords in Czech.

Abstract

Summarize the contents and contribution of your work in a few sentences in English language.

Keywords Replace with comma-separated list of keywords in English.

Contents

List of Figures

Introduction

The first chapter gives an introduction to Vehicle Routing Problem(VRP) and describes motivation, why should we study it. We also outline basic definition and notation of the problem. In the last section are mentioned and reviewed available open-source solvers, to which we later compare our results.

Motivation and objectives

Vehicle routing problem has been extensively studied since 1959[1], when was also defined under the name of Truck Dispatching Problem. The problem appears in many forms in real life and solution to the problem can save companies a lot of money. It's an important problem in the fields of transportation, distribution, and logistics.

The problem can be simply describe as: Having a fleet of vehicles, number of customers and depot, each vehicle starts from a depot, visits some customers and returns back to the depot. At the end all customers should be served and the objective function should be minimal. It's an combinatorial optimization and integer programming problem and the goal is to minimize an objective function, which can vary. The problem belongs to the class of NP-Hard problems, therefore for large real world problems isn't applicable an exact solution.

For specific variant of the problem known as Capacitated Vehicle Routing Problem(CVRP), we have implemented some of the known algorithms, developed their variants and applied to them different strategies. We implement two strategies, which are based on local search and haven't been well described in literature, and we apply them to CVRP obtaining very good results in a little time.

Definition

The definitions of the problem vary depending on the constraints, graph and objective function. In literature and it's simplest variant CVRP, the problem is mostly defined on a complete undirected graf $G = (V, E)$, where V is a set $(0, ..., N)$ of vertices. Each vertex i excluding vertex 0 represents a customer and it's demand q_i. $N - 1$ is the number of customers. Vertex 0, where starts and ends every route R_i is called depot. E is a set of edges $e_{ij}(i, j \in V)$ and every edge e_{ij} has a cost c_{ij}. A fleet of vehicles m, where each vehicle has defined identical capacity Q waits at the depot. We have to determine a number of vehicles m and their routes R_i $(i \in (0, m))$ such that:

- The total demand of route doesn't exceed the vehicle capacity
- Each customer is visited only once by exactly one vehicle

The solution consists of a set of m cycles sharing the depot vertex 0. In many variants of the problem the common objective is to minimize the number of vehicles m and the total cost of routes in set R.

VRP variants

Over more than 50 years there has been described many variants of the classical problem. We will just mention few of them to show how wide and complex the problem is. Between the most studied belong:

Capacitated Vehicle Routing Problem(CVRP)
: The problem has been defined in previous section and we can find it in real-life dispatching products problems.

Vehicle Routing Problem with Time Windows(VRPTW)
: VRPTW is a variant, where the objective is to serve the demands of customers in predefined time windows. There has been variants with soft windows allowed(we can deliver in different time with some penalty).

Vehicle Routing Problem with Pick-Up and Delivery(VRPPD)
: VRPPD is similar to CVRP, but we have two types of items. Ones to deliver to customer from depot and the other to deliver from customer to depot.

Other studied variants are for example:

- dynamic - some customers aren't known in advance and are added in real-time
- heterogenous - there exist different types of vehicles with different capacities

- multiple depots - route can start in any of existing depots and end there, variants with vehicles corresponding to specified depot
- periodic - we need to solve the problem for m days
- split delivery - more than one vehicle can visit one customer
- stochastic - one or more components are random(customer i presented with probability p_i, random demands, random cost c_{ij},...). First solution is generated before knowing the values and second solution corrects that solution after knowing the random variables
- with backhauls - similar to VRPPD, but with a restriction, that all demands on route r_i has to be completed before any pick-ups are made on that route
- with satellite facilities - these can replenish a vehicle, so that the vehicle can continue until specified time

Because of the complex structure of the real-life problems, there exists numerous combinations of these and other variants.

Free/Open-source solvers

There has been few free/open-source solvers, which can solve different types of VRP with different number of customers. For CVRP we present at the end of this chapter results obtained from these algorithms and a comparison with our solution on benchmark instances. Now we present the summary of these solvers(we have excluded some, which implement only basic heuristics) and their brief description obtained from the cited source with some extra findings.

jsprit[2] Jsprit is a java based, open source toolkit for solving rich traveling salesman (TSP) and vehicle routing problems (VRP). It's core meta-heuristic algorithm uses ruin-and-recreate principle(destroy part of the solution and recreate that differently).

Open-VRP[3] Open VRP is a framework to model and solve VRP-like problems for students, academics, businesses and hobbyist alike. It includes implementation of greedy heuristics and tabu search.

OptaPlanner[4] OptaPlanner is a constraint satisfaction solver written in Java. It supports many different problems and includes their examples.

SYMPHONY[5] SYMPHONY is an open-source solver for mixed-integer linear programs (MILPs) written in C. It can be also executed in parallel.

VRP Spreadsheet Solver[6] The Microsoft Excel workbook VRP Spreadsheet Solver is an open source unified platform for representing, solving, and visualising the results of VRP. It unifies Excel, public GIS and metaheuristics. It can solve VRP with up to 200 customers.

VRPH[7] VRPH is an open source library of heuristics for the capacitated Vehicle Routing Problem (VRP). It includes several example applications that can be used to quickly generate good solutions to VRP instances containing thousands of customer locations. It has been developed as part of Chris Groer's dissertation while at the University of Maryland with advisor Bruce Golden.

Vroom[8] Vroom contains different libraries and frameworks for solving VRP and dynamic VRP.

Capacitated Vehicle Routing Problem

Introduction

CVRP is one of the oldest and most studied variants of the problem. Over the time there has been invented and applied many algorithms. We have concentrated on meta-heuristics, especially the ones, that use local search strategies. We first define a mathematical model of the problem as described in[9], then we we briefly overview the methods used to solve the problem. We introduce used benchmarks, implemented strategies and algorithms. At the end of this chapter we present a comparison of the implemented algorithms, comparison with best known results and with available open-source solvers.

Definition

In the first chapter we have already described the problem, but for completeness we provide an integer programming formulation that has been described in[9]:

As previously defined: N ... number of vertices N-1 ... number of customers vertex 0 ... depot c_{ij} ... cost of traversing edge (i,j) Q ... capacity of each vehicle R ... set of routes in final solution m ... number of vehicles in final solution

min

$$\sum_{i=0}^{N}\sum_{j=0}^{N} c_{ij} * x_{ij}$$

subject to:

$$\sum_{j=1}^{N} x_{0j} = 2m$$

$$\sum_{i \leq k} x_{ik} + \sum_{j > k} x_{kj} = 2(k \in (1, N))$$

Solution methods

Exact algorithms

Because of the NP-Hard nature of the problem, exact algorithms can't solve large instances. They can solve instances with up to 200 customers, but weren't able to solve some instances with even less(for example instance CMT6 with 50 customers hasn't been solved to optimum yet). They are very complex and solved with specialized solvers using mixed-integer linear model formulation. We haven't studied these methods, but mention them here for completness. The two common strategies are:

- Branch and bound - state-space search. Let set of solutions form a tree. The algorithm explores this tree and in every branch checks the lower and upper estimated bounds. If it's not satisfied, the solution is discarded.
- Branch and cut - generalization of branch and bound where, after solving the LP relaxation, and having not been successful in pruning the node on the basis of the LP solution, we try to find a violated cut. If one or more violated cuts are found, they are added to the formulation and the LP is solved again. If none are found, we branch.

To our acquired knowledge from [10], there has been recently published a Branch-Cut-and-Price algorithm(BCP)[11], which is very complex and can solve most instances up to 275 customers. The number of customers, that can solve also depends on Q and the length of the routes. The computational time can vary from minutes to hours.

Heuristics

Meta-heuristics

Benchmarks

There has been many benchmark instances for the problem, most of them having up to 200 customers. For the purpose of testing we have chosen 3 different benchmark sets. The summary and other benchmark sets can be found in[10].

Christofides, Mingozzi and Toth (1979)[12]
: This set contains 14 instances with a range of customers between 50 and 199. They have random structure and in instances 11 to 14 the

customers are clustered in groups. Some of the instances also have a constraint on maximum length of a route. All the instances in this set has been solved to optimum by very sophisticated solvers and most of modern meta-heuristic approaches can solve most of them to optimal value. We have chosen this set, because it has quite small number of customers and some of open-source solvers have been tested on it too. With a little playing with the arguments of our algorithms we were able to solve all instances to optimum.

Li et al. (2005)[13]
This set contains 12 instances with a range of customers between 560 and 1200. It includes 3 largest instances ever published in research papers: 1040, 1120 and 1200 customers. This set has been very popular, because of its size. However the set is extremely artificial and all the customers are located in concentric circles around the depot. The routes in best known solutions have almost identical shapes, or the solution is somehow symmetrical. The optimum hasn't been proved in any of them yet. From all 3 benchmark sets our algorithms had most problems with this one and has on average differ less than 6% from the best known value. We think that the big gap from the best known solution is mainly because of the artificial structure and positioning of the customers. Our algorithms are based on local improvements, which are difficult to make if all neighbours of a customer have similar distances to that customer.

Uchoa et al. (2014)[10]
This is a recently published set, which consists of 100 instances with a range of customers between 100 and 1000. All the customers are generated in random clusters. This new set was designed in order to provide a more comprehensive and balanced experimental setting and is intended to be primarily used in the next years. If we compare our algorithm to the state-of-the-art algorithms compared in [10], it appears, that in some instances we can compete with them and with bigger problem size our running time is much less. More closely we will look on this in the comparison section.

Implemented strategies, neighbourhoods

Local search

Local search is a strategy, which locally improves objective function.
Local optima,.....

Neighbourhood

Neighbourhoods are used in local search methods

Swap

Swap procedure takes two different nodes i, j from $N - \{0\}$ and swaps these nodes.

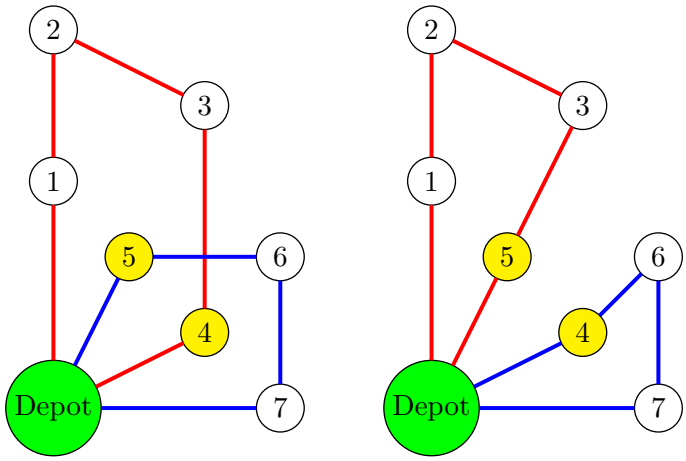


Figure 0.1: Before and after the procedure Swap(4,5)

Reverse edge

Reverse edge procedure takes two different nodes i, j from $N - \{0\}$, which have a common route and reverses the sub-route from i to j including both i and j.

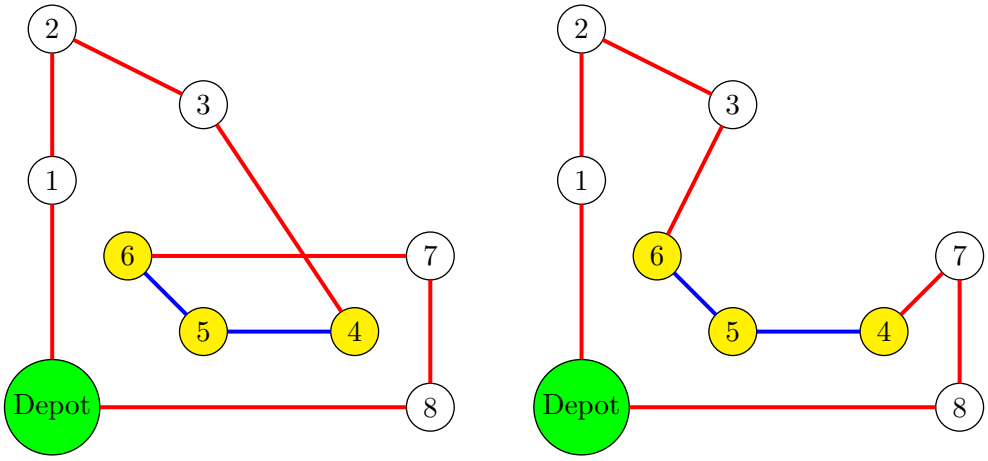


Figure 0.2: Before and after the procedure ReverseEdge(4,5)

Delete and insert

Delete and insert procedure takes a node i from $N - \{0\}$, deletes it and inserts it in the best possible place among the routes.

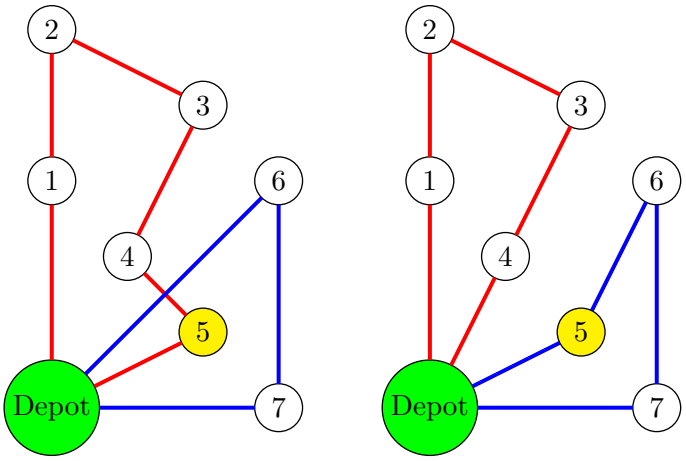


Figure 0.3: Before and after the procedure DeleteAndInsert(4)

Double delete and double insert

Double delete and double insert procedure takes two different nodes i, j from $N - \{0\}$, which are then deleted and reinserted at the best place in the opposite routes. Picture:

First fit

Strategy used in local search algorithms, which picks the first neighbouring solution, that satisfies the constraints. The advantage is, that it usually finds a solution quickly. However, if there are not many feasible solutions it can end up searching through all of them. The disadvantage is, that it picks always the same neighbour.

Best fit

Strategy used in local search algorithms, which tries all the possible neighbouring solutions and picks the best one. The advantage is, that it always picks the best neighbourhood. The disadvantage is, that it always picks the same neighbourhood and has to search all neighbourhoods, which in G costs at least $O(N^2)$ operations.

Random fit

Strategy used in local search algorithms, which randomly chooses the neighbourhood solution, that satisfies the constraints. The advantage is, that it adds randomization to the algorithm and so we have a bigger chance not to end up in the same local minimum. However, if we have little feasible neighbourhoods left, it can take more time to find the feasible one.

Closest search

Strategy used in local search algorithms, which searches neighbouring solution only among closest neighbours.

Implemented algorithms

Savings algorithm[14]

Algorithm 1 Savings algorithm

```
1:  procedure algorithmSavings
2:      Initialize each customer in his own route
3:      for i = 1; i ≤ K; i ← i + 1 do
4:          for j = i + 1; j ≤ K; j ← j + 1 do
5:              s[it].cost ← d[i][0] + d[0][j] − d[i][j]
6:              s[it].i ← i
7:              s[it].j ← j
8:              it ← it + 1
9:      Sort s in descending order
10:     for all x in s do
11:         a ← x.i
12:         b ← x.j
13:         if ConstraintsViolated(a,b) then
14:             continue
15:         if a.next = depot AND b.next = depot then
16:             reverseRoute(b)
17:         if a.next = depot AND b.prev = depot then
18:             connectRoutes(a,b)
```

Hill-Climbing algorithm

Algorithm 2 Hill-Climbing algorithm

```
1:  procedure algorithmHillClimb
2:      S ← feasible solution
3:      C(S) ← cost(S)
4:      while 1 do
5:          S* ← BestFeasibleNeighbourOf(S)
6:          C(S*) ← cost(S*)
7:          if C(S*) łC(S) then
8:              S ← S*
9:              C(S) ← C(S*)
10:         else
11:             break
```

Late Acceptance Hill-Climbing algorithm

Algorithm 3 Late Acceptance Hill-Climbing algorithm

```
1: procedure algorithmLAHC
2:     S ← feasible solution
3:     C(S) ← cost(S)
4:     lastCosts[] ← C(S)
5:     n ← 0
6:     while Stopping criteria not met do
7:         S* ← FeasibleNeighbourOf(S)
8:         C(S*) ← cost(S*)
9:         if C(S*) ≤ C(S) OR C(S*) ≤ lastCosts[nmodpL] then
10:            S ← S*
11:            C(S) ← C(S*)
12:        lastCosts[nmodpL] ← C(S)
13:        n ← n + 1
```

Step Counting Hill-Climbing algorithm

Algorithm 4 Step Counting Hill-Climbing algorithm

```
1:  procedure ALGORITHMSCHC
2:      S ← feasible solution
3:      C(S) ← cost(S)
4:      B ← C(S)
5:      n ← 0
6:      while Stopping criteria not met do
7:          S* ← FeasibleNeighbourOf(S)
8:          C(S*) ← cost(S*)
9:          if C(S*) ≤ C(S) OR C(S*) ≤ B then
10:             S ← S*
11:             C(S) ← C(S*)
12:         if pL ≤ n then
13:             B ← C(S)
14:             n ← 0
15:         n ← n + 1
```

Simulated Annealing

Algorithm 5 Simulated Annealing algorithm

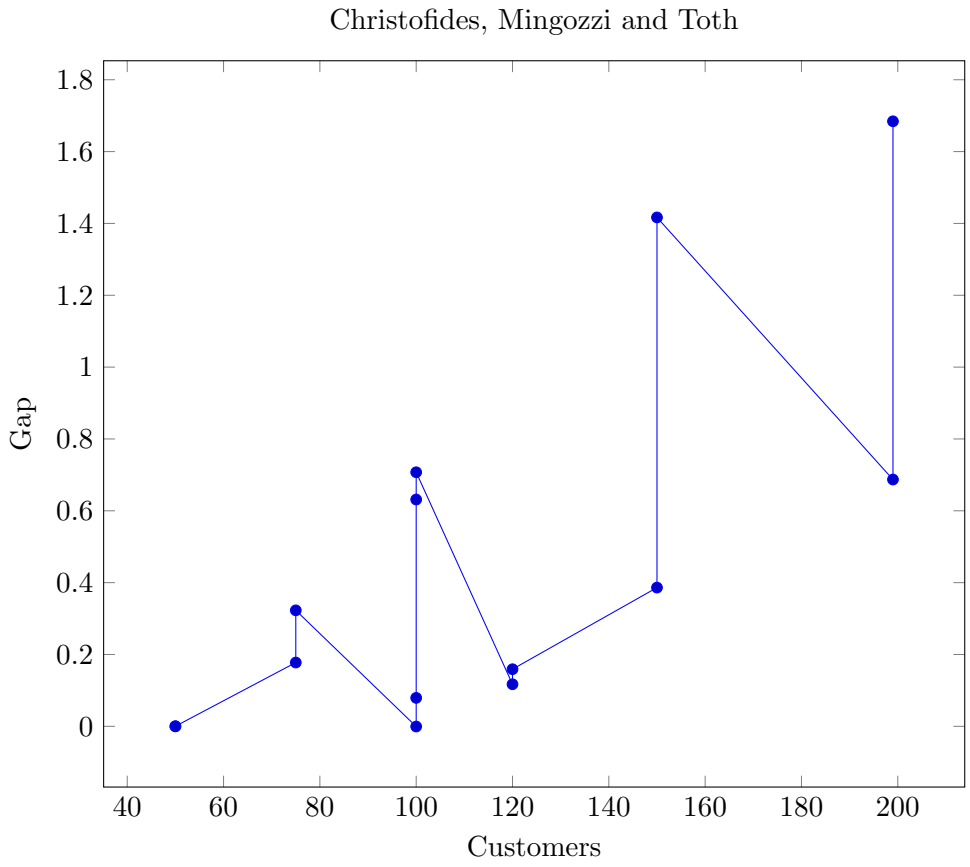
```
1:  procedure algorithmSimulatedAnnealing
2:      S ← feasible solution
3:      C(S) ← cost(S)
4:      while eT ≤ T do
5:          S* ← FeasibleNeighbourOf(S)
6:          C(S*) ← cost(S*)
7:          if C(S*) ≤ C(S) OR random(0,1) ≤ exp((C(S) − C(S*))/T) then
8:              S ← S*
9:              C(S) ← C(S*)
10:         T ← T * fT
```

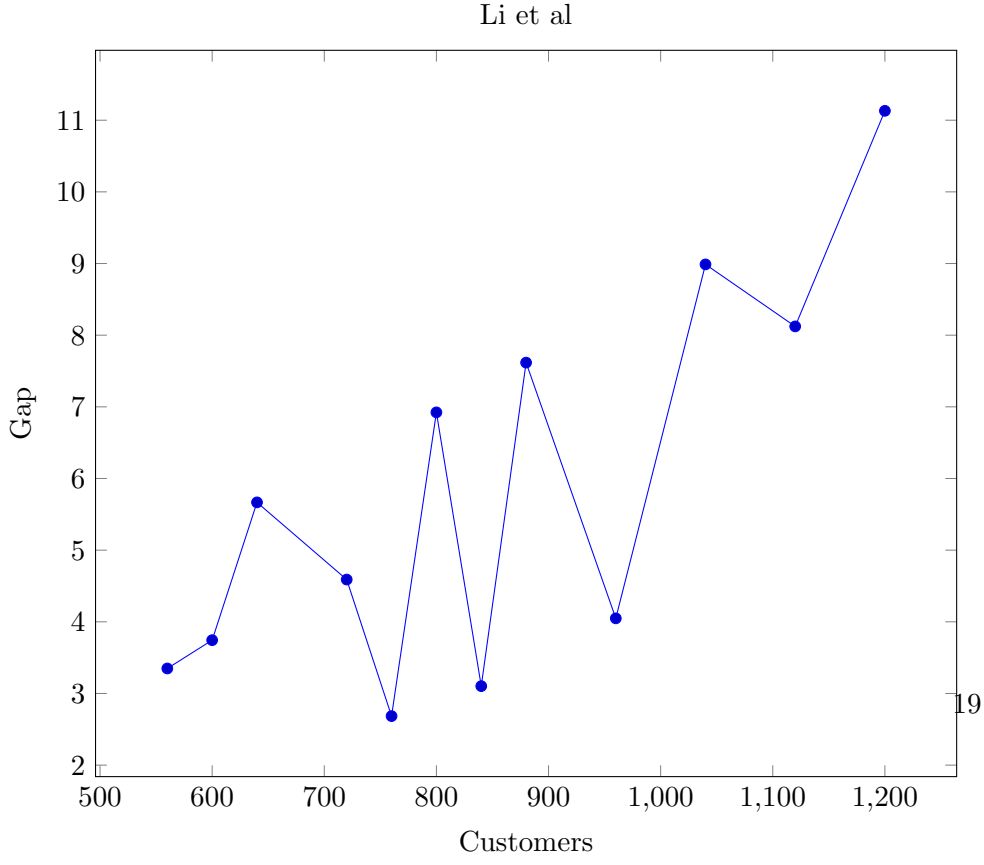
Tabu Search

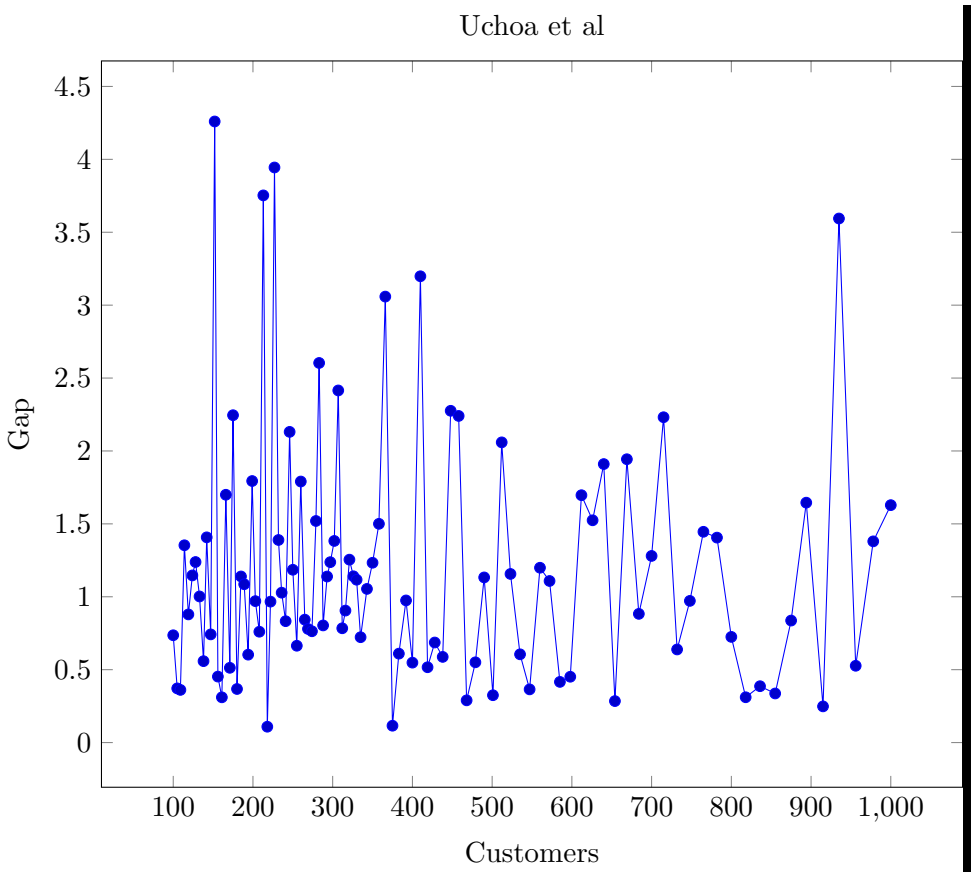
Comparison of implemented algorithms

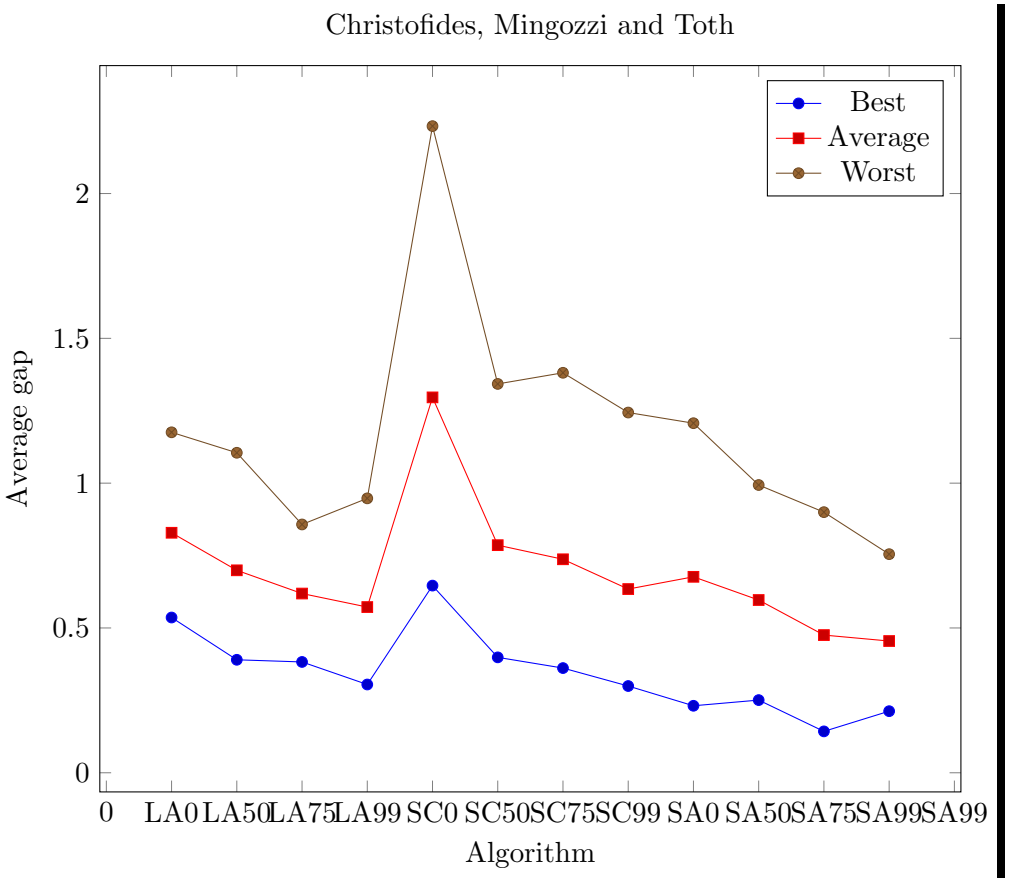
We have compared the implemented algorithms on different benchmark instances. Because it is difficult to compare these algorithms and most of them has parameters, which work best when found experimentally on a specific instance of the problem, we run the algorithms with the same algorithms, so we can also compare how the algorithms behave in general. We have allowed a maximum time of 5 minutes for one instance. However some of the algorithms running time is few seconds. With bigger size of the problem the running time is also bigger. In instances with many customers(¿ 500), the results aren't as close to the best known as in smaller instances, because the algorithms would need more time to search the neighbourhoods.

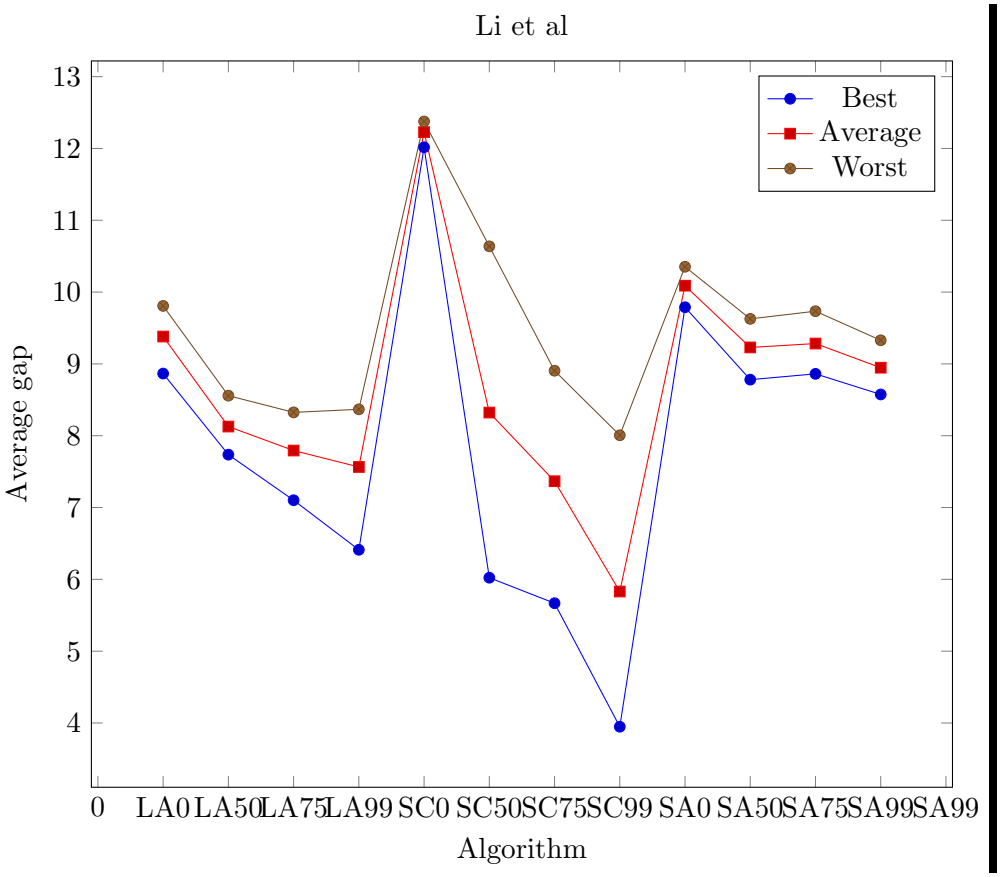
Comparison with best-known results

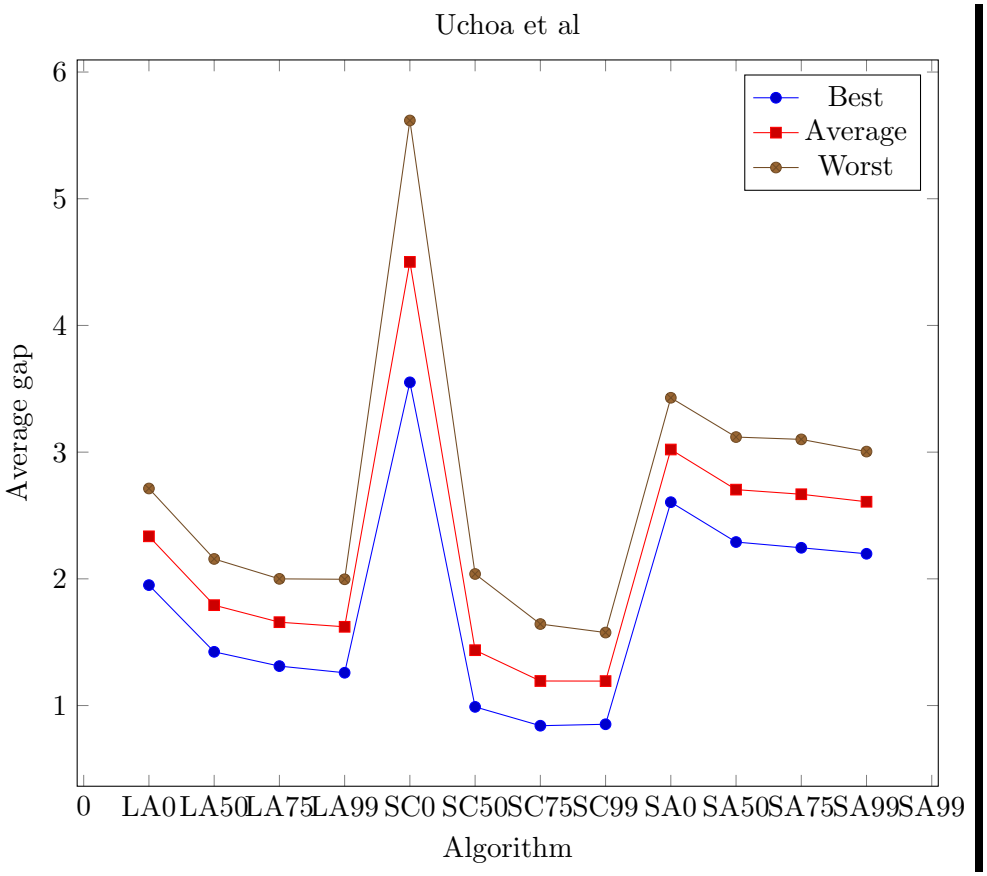





Uchoa et al
Gap
Customers
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
100
200
300
400
500
600
700
800
900
1,000


Christofides, Mingozzi and Toth
Best
Average
Worst
Average gap
0
0.5
1
1.5
2
0
LA0
LA50
LA75
LA99
SC0
SC50
SC75
SC99
SA0
SA50
SA75
SA99
SA99
Algorithm


Li et al
Best
Average
Worst
Average gap
13
12
11
10
9
8
7
6
5
4
0
LA0
LA50
LA75
LA99
SC0
SC50
SC75
SC99
SA0
SA50
SA75
SA99
SA99
Algorithm



Comparison with open-source solvers

Conclusion

Conclusion

Bibliography

[1] Dantzig, G. B.; Ramser, J. H. The Truck Dispatching Problem. *Management Science*, volume 6, no. 1, 1959: pp. 80–91.

[2] Schröder, S. jsprit. Available from: `http://jsprit.github.io/`

[3] Open-VRP. Available from: `https://github.com/mck-/Open-VRP`

[4] Smet, G. D. OptaPlanner. Available from: `http://www.optaplanner.org/`

[5] SYMPHONY. Available from: `https://projects.coin-or.org/SYMPHONY`

[6] Erdoğan, G. VRP Spreadsheet Solver. 2013. Available from: `http://verolog.deis.unibo.it/vrp-spreadsheet-solver`

[7] Groer, C. VRPH. Available from: `http://www.coin-or.org/projects/VRPH.xml`

[8] Pillac, V. Vroom. Available from: `http://victorpillac.com/vroom/`

[9] Fisher, M. L.; Jaikumar, R. A generalized Assignment heuristic for Vehicle Routing. 1979.

[10] Uchoa, E.; Pecin, D.; Pessoa, A.; et al. New Benchmark Instances for the Capacitated Vehicle Routing Problem. 2014.

[11] Pecin, D.; Pessoa, A.; Poggi, M.; et al. Improved branch-cut-and-price for capacitated vehicle routing. In *Integer Programming and Combinatorial Optimization*, Springer, 2014, pp. 393–403.

[12] Christofides, N.; Mingozzi, A.; Toth, P. The vehicle routing problem. *Combinatorial Optimization*, volume 1, 1979: pp. 315–338.

[13] Li, F.; Golden, B.; Wasil, E. ery large-scale vehicle routing: new test problems, algorithms, and results. *Computers & Operations Research*, volume 32, no. 5, 2005: pp. 1165–1179.

[14] Clarke, G.; Wright, J. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, volume 12, no. 4, 1964: pp. 568–581.

Appendix A

Acronyms

VRP Vehicle routing problem

CVRP Capacitated vehicle routing problem

VRPTW Vehicle routing problem with time windows

CCVRP Cummulative capacitated vehicle routing problem

LAHC Late Acceptance Hill-Climbing algorithm

SCHC Step Counting Hill-Climbing algorithm

SA Simulated Annealing

TS Tabu Search

Appendix B

Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `exe` ..... the directory with executables
- `src` ..... the directory of source codes
  - `wbdcm` ..... implementation sources
  - `thesis` ..... the directory of LaTeX source codes of the thesis
- `text` ..... the thesis text directory
  - `thesis.pdf` ..... the thesis text in PDF format
  - `thesis.ps` ..... the thesis text in PS format