

# IFT6751: Homework 2

Gabriel C-Parent

March 19, 2015

## 1 Introduction

In this homework, two different approaches to solve the capacitated vehicle routing problem (CVRP) were designed. The first one is a genetic algorithm using specialized crossover and mutation operators whereas the second one uses the Tabu Search to control local search with the  $\lambda$ -interchange neighbourhood.

Two greedy local search methods are used within both methods to improve the results such as the 2-opt descent and the  $\lambda$ -interchange descent.

Another method based on the Clark & Wright savings is also used to initialize solutions for both metaheuristics.

Both methods were tested against problem instances without length limit, from [1] and compared to the best known solutions.

A description of all the optimization methods along with special implementation details is given. Some experimental results are then compared based on running time, implementation complexity and results quality.

## 2 Local search methods

### 2.1 2-opt descent

First, a simple and fast optimization method for the TSP was needed to improve the path of each routes.

The local search method used to optimize individual routes is the steepest improvement method as described in [4]. Basically, it is a greedy use of the well known 2-opt method.

At each iteration, the best possible 2-opt is chosen, according to the reduction in total distance, until there isn't any possible improvement. The complexity of the procedure  $O(n^2)$  on the number of edges in the route.

Although it might seem slow, usually the number of edges is quite small and the time spent optimizing routes is relatively negligible.

Simplified python code can be seen in the code section 9.1.

### 2.2 $\lambda$ -interchange descent

Another optimization method was needed to exchange clients between routes, in this case the  $\lambda$ -interchange descent with  $\lambda=1$  [3].

The possible transitions are the transfer of a client from a route to another and the swap of two clients between two routes. Since only feasible solutions are considered, only transitions that do not violate the capacity limit are considered.

The procedure chooses the interchange with best possible improvement and applies it. Then the 2-opt descent is applied to the modified routes and the process is repeated until it gets stuck in a local minima.

### 3 Random Savings Initialization

The Clark & Wright savings algorithm is a well known simple heuristic for the CVRP. Many improvements were suggested for this heuristic [2].

The one used in this work is a slight variant of the parallel savings, where instead of choosing the best the best saving and merging the corresponding routes, the  $k$  best savings are found and one is randomly chosen.

This procedure is used in both the Tabu Search and Genetic Algorithm procedures.

In the Genetic Algorithm procedure, the random savings is used to generate good initial solutions. The initialization step is costly but the quality of the initial population is great.

In the Tabu Search procedure, the random savings is used to initialize a solution, which is then locally explored until convergence is achieved.

## 4 Genetic Algorithm

### 4.1 Encoding

The solutions are encoded using the Route object. Basically, a route is a list of clients that starts at the depot and ends at the depot, where no client is repeated. The solution is a list of routes, where each client is in exactly one route.

This representation isn't really friendly to classical crossover operators but allows functions defined on routes and solutions to be shared for both the Tabu Search and Genetic Algorithm representations.

### 4.2 Objective function

The objective function is the minimization of the total distance of the routes.

$$\begin{aligned} &\text{minimize} && \sum_{r \in \text{solution}} \text{distance}(r) \\ &\text{subject to} && \text{weight}(r) \leq \text{vehicle capacity} \end{aligned}$$

### 4.3 Selection

The simple well known binary tournament selection is used to select the parents of the next generation.

### 4.4 Crossover

The crossover used here is related to the concept of petals. Basically, the

### 4.5 Mutation

### 4.6 Population swap

### 4.7 Stopping criteria

## 5 Tabu search

With minor variations, the Tabu Search implemented was the same as in [3].

### 5.1 Solution space

Only feasible solutions are considered, that is those not violating the capacity constraint.

### 5.2 Objective function

The objective function is the minimization of the total distance of the routes.

$$\begin{aligned} & \text{minimize} && \sum_{r \in \text{solution}} \text{distance}(r) \\ & \text{subject to} && \text{weight}(r) \leq \text{vehicle capacity} \end{aligned}$$

### 5.3 Neighbourhood of a solution

The neighbourhood of a solution is all the feasible solutions that can be reached by applying the  $\lambda$ -interchange, with  $\lambda=1$ .

### 5.4 Tabu list

### 5.5 Aspiration criteria

### 5.6 Intensification

### 5.7 Diversification

The diversification strategy used here is to restart the Tabu Search with a new solution created by random savings.

### 5.8 Stopping criteria

As with the genetic algorithm, the stopping criteria can be a set number of iterations or a time limit, depending on the type of constraints the user has.

## 6 Experimental results

## 7 Analysis of experimental results

### 7.1 Genetic Algorithm

### 7.2 Tabu Search

### 7.3 Non-dominated treatments

## 8 Conclusion

## References

- [1] N Christofides, A Mingozi, P Toth, and C Sandi. Combinatorial optimization. 1979.
- [2] M D Nelson, K E Nygard, J H Griffin, and W E Shreve. Implementation techniques for the vehicle routing problem. *Comput. & Ops. Res.*, 12(3), 1984.
- [3] I H Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 41, 1993.
- [4] Áslaug Sóley Bjarnadóttir. Solving the vehicle routing problem with genetic algorithms. Master's thesis, Technical University of Denmark, 2004.

## 9 Code example

### 9.1 2-opt descent

---

```
1 def steepest_improvement(route, dist):
2     """steepest improvement procedure, using 2-opt"""
3     best_ind1 = 0
4     best_ind3 = 0
5     savings = 0.
6     proposed_savings = 0.
7     # iterate until there isn't any better local choice (2-opt)
8     while True:
9         savings = 0.
10        for ind1 in range(0, len(route.nodes)-2):
11            for ind3 in range(ind1+2, len(route.nodes)-1):
12                n1 = route.nodes[ind1]
13                n2 = route.nodes[ind1 + 1]
14                n3 = route.nodes[ind3]
15                n4 = route.nodes[ind3+1]
16                actual = dist[n1][n2] + dist[n3][n4]
17                proposed = dist[n1][n3] + dist[n2][n4]
18                proposed_savings = actual - proposed
19                if proposed_savings > savings:
20                    best_ind1 = ind1
21                    best_ind3 = ind3
22                    savings = proposed_savings
23            if savings > 0.:
24                # do the 2-opt exchange
25                two_opt(route, best_ind1, best_ind3)
26            else:
27                return
28    return
```

---