IFT6751: Homework 2

Gabriel C-Parent

March 15, 2015

1 Introduction

In this homework, two different approaches to solve the capacitated vehicle routing problem (CVRP) were explored. First, a genetic algorithm procedure was devised, using Clark & Wright savings [1] to A simple Tabu Search heuristic was implemented, based roughly on along with a Genetic Algorithm.

Both methods were tested against problem instances without length limit, from [2] and compared to the best known solutions.

A description of each algorithm is given and then some experimental results are compared.

2 Local search method

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.

3 Clark & Wright Random Savings

The Clark & Wright savings algorithm is a well known simple heuristic for the CVRP. Many improvements were suggested for this heuristic [3].

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 Selection

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

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] I K Altinel and Oncan T. A new enhancement of the clarke and wright savings heuristic for the capacitated vehicle routing problem. *The Journal of the Operational Research Society*, 56(8), 2005.

[2] N Christofides, A Mingozzi, P Toth, and C Sandi. Combinatorial optimization. 1979.

[3] M D Nelson, K E Nygard, J H Griffin, and W E Shreve. Implementation techniques for the vehicle routing problem. 1984.

[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 Steepest Improvement

```
def steepest_improvement(route, dist):
    """steepest improvement procedure, using 2-opt"""
    best_ind1 = 0
    best_ind3 = 0
    savings = 0.
    proposed_savings = 0.
    # iterate until there isn't any better local choice (2-opt)
    while True:
        savings = 0.
        for ind1 in range(0, len(route.nodes)-2):
            for ind3 in range(ind1+2, len(route.nodes)-1):
                n1 = route.nodes[ind1]
                n2 = route.nodes[ind1 + 1]
                n3 = route.nodes[ind3]
```

```
                n4 = route.nodes[ind3+1]
                actual = dist[n1][n2] + dist[n3][n4]
                proposed = dist[n1][n3] + dist[n2][n4]
                proposed_savings = actual - proposed
                if proposed_savings > savings:
                    best_ind1 = ind1
                    best_ind3 = ind3
                    savings = proposed_savings
        if savings > 0.:
                # do the 2-opt exchange
            two_opt(route, best_ind1, best_ind3)
        else:
            return
    return
```