A Three Pronged Approach to CVRP Based on the Clarke & Wright Savings Algorithm

Chris Birmingham | Bristol Robotics Laboratory | cb16869@bristol.ac.uk

Introduction.

The Capacitated Vehicle Routing Problem is a type of problem requiring the delivery of a specified amount of product to a set of locations from a supply depot using a vehicle which can only carry a limited quantity of said product. The problem is similar to the Traveling Salesman Problem in that the objective is to optimize the routes taken to minimize the distance traveled, but with added constraints.
This problem is a NP-Hard problem for which there is no known method for obtaining the optimal solution to non-trivial sized problems. For the sample problem assigned with 249 customers, there are at least 249! (3.2E493) valid solutions. To compute each of these solutions and find the optimal (shortest) solution is impossible in the human lifetime. Despite the complexity of finding the perfect solution, with heuristics it is possible to find good solutions to even large problems like the sample problem assigned. The quality of these solutions is important, as there are many CVRP type problems in the real world for which a solution can save time and money.

Implementation.

The solution proposed here (as seen in Figure 1) is a modified form of the Clarke-Wright Savings Algorithm with a genetic algorithm applied to the savings list and 2-opt swap applied to the final vehicle routes. Alternatively, this solution could be described as a genetic algorithm in which the initial population of chromosomes is made up of shuffled Clarke-Wright savings lists and the fitness function is the distance produced by creating a set of routes with the Parallel Savings Algorithm and then optimized with 2-Opt swap.
To produce the savings matrix in Figure 1, the distance saved when two customers are combined onto the same route (Figure 2), is calculated with Equation 3. The savings matrix is then sorted into a sequential list of pairs, with the highest savings on top. Here, instead of using the Parallel Savings Algorithm as in the traditional CW Savings Algorithm, the list is then segmented into genes that are 50 pairs long (Figure 3). A starting population for the GA is produced by shuffling the ordering of the pairs in each gene for each new chromosome. From here the GA begins the process of optimizing the chromosomes for the shortest distance, evaluated by the fitness function. The fitness function in this case is the distance produced when the pairs are used to create a set of routes that are then optimized with a greedy 2-Opt swap. After there is no more improvement between rounds the GA outputs the resulting list of routes to a text file and produces a map of the final set of routes (Figure 4).

Pseudocode.

```
def main():
    location_dictionary = read_file(raw_input)                  #read text file

    savings_matrix = create_savings_matrix(location_dictionary)     #for every pair calculate the savings if joined together
    sorted_savings_list = create_sorted_savings_list(savings_matrix)#sort the pairs from highest to lowest savings
    generation = chromosome_generation( num_chromosomes, allele_size, sorted_savings_list)# create the first generation of chromosomes
                                                                #by randomly shuffling the order of pairs in each gene
    best_route, best_distance, second_best_route = evaluate_chromosomes( generation )#Evaluate chromosomes and store the best result
    best_score_so_far = best_distance
    best_route_so_far = best_route
    while improvement:                                          #Continue GA while improvements continue
        generation = reproduce_chromosomes(best_route, second_best_route)
        best_route, best_distance, second_best_route = evaluate_chromosomes( generation )
        if best_distance > best_score_so_far:
            best_route_so_far = best_route
            best_score_so_far = best_distance
            improvement = True
        else:
            improvement = False
    write_out_best(best_route_so_far, best_score_so_far)
    plot_data(best_route_so_far, best_score_so_far)
    return()
```

Conclusion.

After attempting to devise a unique solution without referencing the literature, I decided to gauge my own algorithm against a popular algorithm from the literature. I chose to use the Clarke Wright Savings Algorithm as my measuring stick because it was described as easy to implement and fast to produce relatively good results [1]. To guide my implementation I used the guides found in [2] and [3]. Checking it against my own algorithm I realized it had produced significantly better results in a fraction of a second. From the literature I knew it was not the perfect solution, so I set about trying to improve it. Looking at the output I knew that the individual delivery routes were imperfect because they were crossing over themselves. To fix this I decided to add a greedy 2-opt swap to optimize the individual traveling salesman problems. I then noticed that the while the ordered savings list generated relatively good routes directly from the Parallel Savings Algorithm, reordering the list could produce better results. To find the best order of the list I settled on using a relatively simple genetic algorithm, based what I learned in class.
Combining the three techniques of the GA, the 2-opt swap, and the CW Savings Algorithm produced better results than any of them would on their own. If computational time is of greatest importance the CW algorithm is the best choice, and if finding the true optimum is the desired goal then another approach may be better, however in terms of simplicity to implement and speed of use, this combination is the best I have found.

References.

[1] Laporte, Gilbert. "The vehicle routing problem: An overview of exact and approximate algorithms." European Journal of Operational Research 59.3: 345-358, 1992.
[2] Caccetta, L., Alameen, M. and Abdul-Niby, M., “An improved Clarke and Wright algorithm to solve the capacitated vehicle routing problem.” Engineering, Technology & Applied Science Research, 3(2), pp.pp-413, 2013.
[3] Lysgaard, J. “Clarke & Wright’s Savings Algorithm.” Department of Management Science and Logistics, The Aarhus School of Business, 44. 1997.

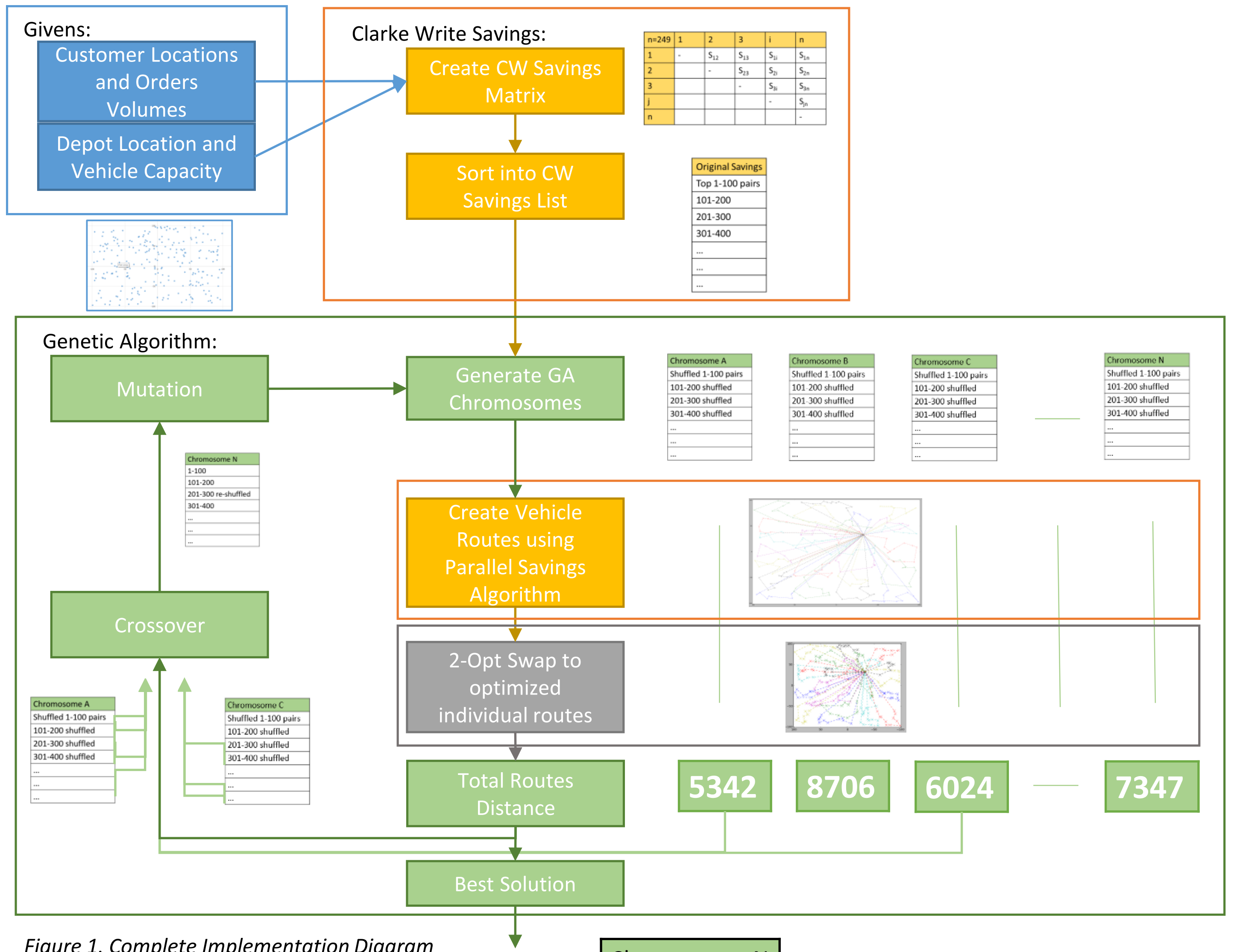
Givens:
Customer Locations and Orders Volumes
Depot Location and Vehicle Capacity
Clarke Write Savings:
Create CW Savings Matrix
Sort into CW Savings List
Genetic Algorithm:
Mutation
Generate GA Chromosomes
Create Vehicle Routes using Parallel Savings Algorithm
2-Opt Swap to optimized individual routes
Crossover
Total Routes Distance
5342
8706
6024
7347
Best Solution

Figure 1. Complete Implementation Diagram

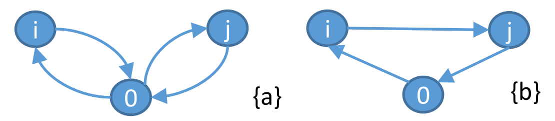

Figure 2. Illustrating the savings of combining two routes into one

$D_a = C_{0i} + C_{i0} + C_{0j} + C_{j0}$ *Equation 1.*

$D_b = C_{0i} + C_{ij} + C_{j0}$ *Equation 2.*

$S_{ij} = D_a - D_b = C_{j0} + C_{0j} - C_{ij}$ *Equation 3.*

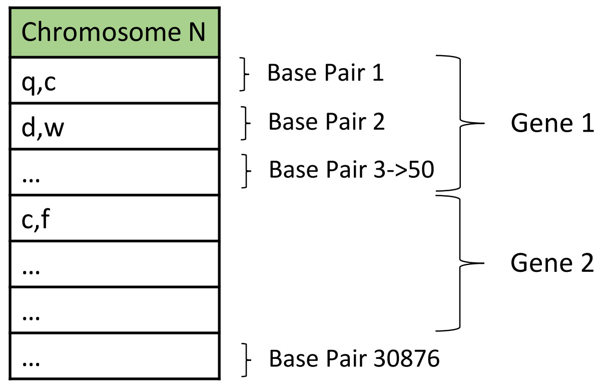

Figure 3. Chromosomes made of all possible pairs of customers. Genes are 100-pair segments of the chromosome

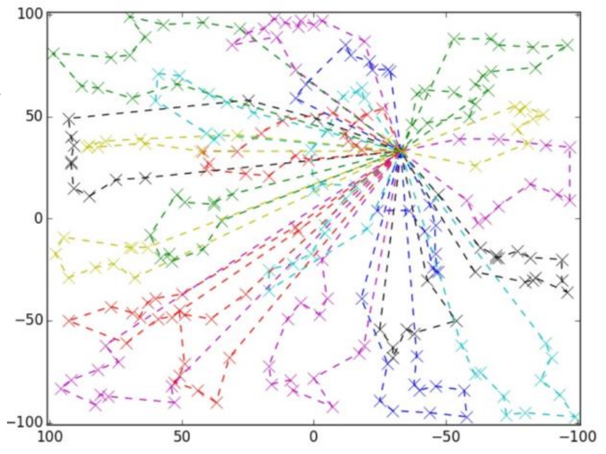

Figure 4. Final Solution, cost: 5685.051

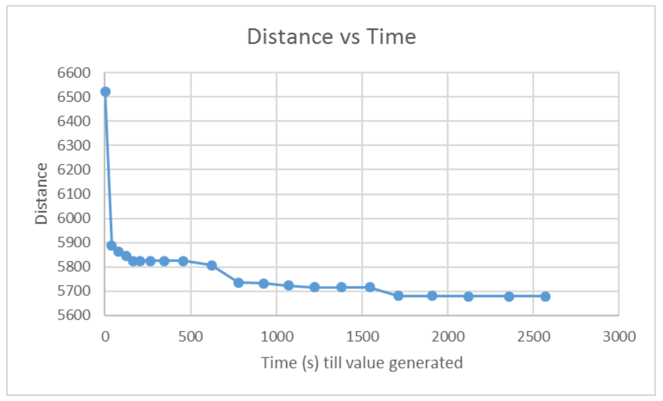

Figure 5. Final solution run, cost: 5685.051