Multi-Depot Vehicle Routing Problem

LIWEN DAI, 20552153

In this project I studied techniques to solve the NP-hard Multi-Depot Vehicle Routing Problem (MDVRP). Details of five algorithms were analyzed: sweep heuristic, tabu search, fast look-ahead, ant colony optimization, and coevolutionary algorithm. Their benchmark results from literature were compared. In the end I discussed which algorithm may be the best choice to solve MDVRP.

1 INTRODUCTION

The Vehicle Routing Problem (VRP) is a generic name which refers to a range of combinatorial optimization problems in physical distribution and logistics systems. Everyday, a delivery company needs to deliver goods to a lot of customers. A fleet of vehicles are dispatched from a centralized storage warehouse and visit customers to achieve delivery. The goal of the problem is to design routes of vehicles, similar to traveling salesman tours, so that each customer is served once by exactly one vehicle, and the total travel distance of vehicles is minimized.

In this project I studied techniques for solving the Multi-Depot Vehicle Routing Problem (MDVRP), a well-known variant, or generalization of VRP. As the name suggests, there are multiple depots available to provide goods, instead of one. Also, each vehicle leaves a depot, visits customers and returns to the same depot, which is close to real-world situation. Additionally, several constraints bring additional complexity: the customers have different demands, and each vechile has a fixed storage capacity for goods, as well as a limit of maximum travel distance. Moreover, each depot potentially has a limited number of vechicles in reserve.

1.1 Problem Definition

Although the exact formulations may vary from researchers, MDVRP can be described as follows according to [11]. Let $G = (V, E)$ be a complete graph, where V is the vertex and E is the edge set. The vertex set V is partitioned into two subsets, n customers $V_c = \{v_1, ..., v_n\}$, and m depots $V_d = \{v_{n+1}, ..., v_{n+m}\}$. Each customer v_i has a positive demand d_i. A travelling cost matrix is defined on the edges, such that $c_{i,j}$ denotes the travelling cost from vertex i to j, where i, j can be either a customer or a depot. The MDVRP requires to construct a set of vehicle routes such that: (1) each route starts and ends at the same depot, (2) each customer is visited exactly once by one vehicle, (3) the total demand of a route $\leq Q$, (4) the total cost of a route $\leq$ a preset limit L_{max}, (5) the sum cost of all routes is minimized.

It is worth mentioning that, in this definition, it isn't explicited required that all vertices in the graph represent points on a 2D plane, which is however, often the assumption in various algorithms, as we will see below [5, 6]. This also suggests that, the traveling cost $c_{i,j}$ is just the distance between i and j where $c_{i,j} = c_{j,i}$.

[11] assumes that at each depot there are only K_m identical vehicles, another constraint not always used by other algorithms. Actually, it was not even used in its own test benchmarks, even if it was implemented. As a result, although limiting the number of vechiles at each depot does make the problem different, I assume each depot has unlimited number of vechiles.

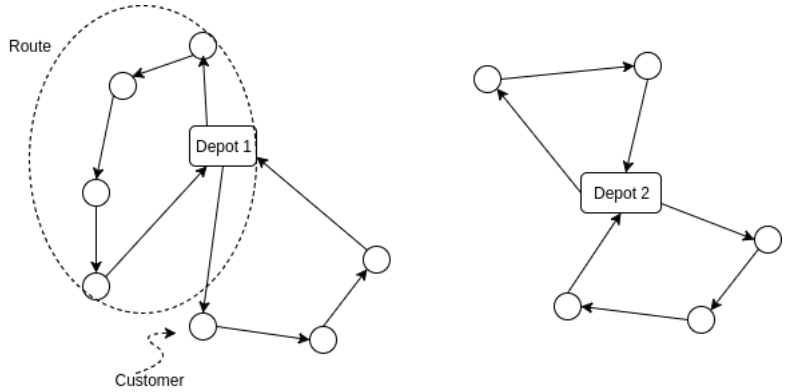

Fig. 1. An example for MDVRP with 2 depots and 12 customers. There are four routes in this example solution, two for each depot.

2 RELEVANT WORK

MDVRP is a generalization of the NP-hard VRP, and is very difficult to solve optimally, even for relatively small problems. One exact approach [7] is formulating the MDVRP as mixed integer programs, and solving them by a branch and bound algorithm. For single-depot capacitated VRP, one of the most successful exact approach, the K-tree method [4] only succeeded in solving a problem with 71 customers. In order to work with larger instances, heuristic methods must be used.

In the next section 5 heuristic approches will be analyzed. The first is the multi-terminal sweep algorithm by Gillett and Johnson [5], a simple heuristic extended from VRP. The second is the tabu search (meta) heuristic [11], which uses smarter search methods and yields better solutions compared to the first one. The 3rd is fast look-ahead heuristic [1], which uses a search tree to predict the best move, and yields good solutions in very short time. The 4th is ant colony optimization [12], which simulates the food-seeking behavours of ant colonies in nature. The 5th is a cooperative coevolutionary algorithms [3], where each subproblem evolves independently by genetic methods.

There are many more works relevant to MDVRP or other VRP variants. By analyzing these five algorithms, my goal is to tell which technique suits best for MDVRP, as well as to summarize general techniques to handle similar complex combinatorial optimization problems.

3 ANALYSIS OF TECHNIQUES

3.1 Multi-Terminal Sweep Algorithm

The Multi-Terminal (depot) Sweep Algorithm is a heuristic algorithm proposed by Gillett and Johnson [5] to obtain an approximate solution to MDVRP. It uses the strategy of partitioning MDVRP in to a set of subproblems, and solving each of them as a single-depot VRP using the sweep heuristic [6]. The initial solution usually requires further refinements and adjustments to achieve near-optimality, since the partitioning process uses a greedy approach and doesn't guarantee optimality itself.

3.1.1 Cluster Partitioning. The proper partitioning of customer points into clusters about individual depots is the key to solving MDVRP. Each cluster consists of a chain of customer points assigned to a single depot, but it is not intended that each chain represents a feasible route. To

build up clusters, unassigned customers are inserted into partially formed clusters, one at a time. The consideration is to minimize the insertion cost for each customer, the extra distance needed to include customer point j between points i and k: $c_{i,j} + c_{j,k} - c_{i,k}$. This determines which depot serves each customer.

3.1.2 Sweep Algorithm for Single-Depot VRP. For each cluster a VRP subproblem is to be solved by the sweep algorithm [6]. Locations are renumbered according to their polar-coordinate angle. The depot has id 1 and the angle is computed by $An(i) = \arctan[(y_i - y_1)/(x_i - x_1)]$ for point i with 2D coordinate (x_i, y_i). As a result, assume $An(i) < An(i+1) \forall i$. The ties are broken by distance: when $An(i) = An(j)$, $i < j$ if $c_{1,i} < c_{1,j}$. The points are further partitioned into routes. The first route consists of customer 2, 3, ..., j where j is the last customer that can be added wihouth exceeding the vechile capacity or distance constraint. The second route starts at $j + 1$ and ends the same way, and so on. The total travel distance is the sum of distances of each route.

The next step if trying to improve by replacing one location in route K with one or more locations in route $K + 1$. The location to be deleted from route K should be close to the the depot as well as to the next route, which is obtained by minimizing the function $R(i) + An(i) \times AVR$, where $R(i)$ is the radius and AVR is the average radii for all locations. The first location, say p, considered to be included in route K from route $K + 1$ is the nearest one to the last location in K. The next considered for inclusion in the nearest to location p, and so on, until no more locations can be included in route K.

The process of replacing locations continues for each route K until no improvement is found. Then the X and Y axes are rotated counterclockwise so the first location becomes the last and the second becomes the first, and so on. Routes partitioning and swapping locations between routes described above is repeated. This rotating process continues until all possibilities are exhusted. The minimum total distance during the process is recorded as the result.

3.1.3 Combining VRP Results. Grouping individual VRP solutions together forms a feasible MDVRP solution, which can be further improved. In Multi-Terminal Sweep Algorithm, this is done by reassigning customers that are in regions "between" depots. Each depot takes a turn to steal customers from nearby depots. When a reassignment is done, the two single-depot VRP must be solved again. However it is not clearly stated how to select candiate customers for the reassignment, which may lead to arbitrary regions in the actual implementation.

3.1.4 Critiques. Sweep heuristic is fast when prototyping the initial solution. The significant computation time is spent in solving individual routes as the Traveling Salesman Problem (TSP). Actually, every time a "check" is made when interchanging locations between routes, two TSPs must be solved to see if this change leads to improvement. If the route consists of small number of locations, the TSP can be solved fast and optimally. In the empirical results of [6], the computation time is linear to the total number of cutomers, when the average number of customers in a route is small. However the time grows quadratically with the average number of customers in a route. This suggests that some $O(n^2)$ method is used to optimize individual TSPs, though the exact approach is not stated.

In following sections we will see smarter approaches to optimize TSP. The sweep algorithm has the problem that it solves TSP too many times. Additionally, it only keeps mutating one state, in addtition to the best solution found so far. This limits its capability to explore the search space. After all, this is a classic algorithm, published in 1976.

3.2 Tabu Search Heuristic

This global optimization metaheursitic strategy for MDVRP is described by [11]. The local search operation explores the search space by moving from a solution to its best neighbour, even if this move doesn't lead to a better objective funtion value. To avoid cycling, states that are visited recently are declared tabu for some iterations, so that they are not revisited.

3.2.1 Initial Solution. Initially each customer is assigned to its nearest depot, and each cluster is solved by a single-depot VRP heuristc. While the sweep heuristic can meet this purpose, a similar approach, the Improved Petal heuristic [10] is used: a set of one-vehicle routes are generated, and the best route selection is done by solving a set partitioning problem. A single-vehicle route is formed as a cycle of envelope (not necessarily convec) for the depot. Remaining customers are inserted to the partial route, which is periodically improved by the 4-opt* edge exchange mechanism [9]. Two-vechile routes are generated only if $n/m \geq 50$ since it can be time consuming.

3.2.2 Search Operators. 4-opt* improvement mechanism [9] for TSP is used to optimize each individual route. Among all potential 2-opt, 3-opt, 4-opt moves, a subset of 8 moves are selected for evaluation. It provides a much richer set of operations compared to the 2-opt, and is more timely efficient than 3-opt exchange procedure. For n points, 4-opt* evaluates only $8wn^2$ moves, where w is a configurable constant. In the experiment, 4-opt* is only 0.95% worse than the standard 3-opt, but uses only 1% of its time when $n = 500$ and $w = 20$.

2-route[11] improvements are obtained by moving vertices belonging to two different routes, around one or two depots. Let (i_1, j_1, k_1, l_1) and (i_2, j_2, k_2, l_2) be two sequences consecutive points on two routes 1 and 2, potentially including depots. Six moves are evaluated, as long as feasibility is maintained and no depot is moved: (1) insert j_1 between i_2 and j_2; (2) swap j_1 and j_2; (3) insert (j_1, k_1) between (j_2, k_2); (4) swap (j_1, k_1) and (j_2, k_2). Also, (1)(3) have their symmetric operations.

3-route exchanges provide more possibility compared to 2-route when routes are nearly full in capacity. Consider a point i_1 in route 1, a point i_2 in route 2 as well as consecutive points (r_2, s_2) in route 2 and (r_3, s_3) in route 3. The combination of moves are evaluated in the feasible domain: insert i_1 between (r_2, s_2); insert i_2 between (r_3, s_3).

3.2.3 Tabu Search. The core this algorithm is called FIND. It consists of three phases: Fast Improvement, Intensifiction and Diversification.

During the fast improvement phase, the following three steps are repeatedly applied: (1) 2-route exchanges between routes of 2 depots; (2) 2-route exchanges between route of the same depot; (3) 3-route exchange. In each step, a move is immediately done if it leads to improvement; otherwise, the best non-tabu deterioating move is performed. Every time a move is done, the 1-route 4-opt* method is used to optimize routes involved in the move.

When selecting which routes to apply the exchange, each route is represented by its center of gravity, in order to calculate distances between routes and depots. Each depot only considers the nearest half of depots. For each depot pair, only half of the routes nearest to the other depot are considered. For intra-depot exchange, all pairs of routes are considered. For 3-route exchange, (h_1, h_2, h_3), h_2 is the closest neighbour route of h_1, and h_3 is the closest neighbour route of h_2. While not much affect to the quality, this filtering significantly reduces the search space.

During the intensification phase, starting at the best solution so far, the two-route exchanges are applied to each depot in turn until no improvement is found for some (300) consecutive iterations. The goal is to intensify the search for better routes of each VRP cluster.

The diversification phase aims to escape the local minimum at inter-depot scope and to perform a broader exploration. Two steps are repeated 20 times. The first step is reinserting a customer into a route of a different depot. The second step is applying the 2-route exchanges of the fast improvement phase, but without improvement to the total distance objective, for 50 consecutive iterations. Additionally, the duration of a tabu move is randomly chosen.

Compared to the sweep algorithm [5], the tabu search algorithm has serveral advantages. First it provides a richer set of local search operators, and a clear description on which move to choose. Next, it maintains a tabu list to avoid revisiting states. Moreover, the diversification phase allows moves that deteriorate the objective function, offering a better exploration in the search space. Last, it provides a flexible configuration of parameters, where the user can control the randomness and running time to some extent. As a result, the tabu search algorithm is easier to use and yeilds better results [11].

3.3 Fast Look-Ahead Heuristic

The Fast Look-Ahead Heuristic, or Look-Ahead and Routing (LAR) [1] also works in stages. It uses a simple construction heuristic to assign customers to depots and uses a savings algorithm to solve the VRP for each cluster. A look-ahead search tree is constructed to approximate the decision quality. Finally it also uses some simple local improvement methods to tune the solution.

3.3.1 Cluster Partitioning. In the first stage, LAR starts by dividing customers into core customers and borderline customers. For a given customer i, this is decided by the ratio of $c_{i,j}/c_{i,k}$, where j is the nearest depot to i, and k is the second-nearest depot to i. If this ratio is smaller than some parameter ϵ, the customer is assigned to the nearest depot as core customer, because the nearest depot is much closer than the second nearest. Otherwise, the customer is left unassigned as borderline customer, since the two closest depots have a similar distance. Next, VRPs are solved for each depot, but only with core customers. Then borderline customers are inserted one by one by decreasing opportunity cost, which is the absolute difference between inserting the customer to a route of the target depot, and that of the closest depot. For a borderline customer i with nearest depot j, all depots k with $c_{i,j}/c_{i,k} > \epsilon$ are considered as potential target depots. The customer is inserted where the insertion cost is minimal. A major difference from other algorithms described above, is that when inserting borderline customers, infeasible routes are allowed with respect to tour length and capacity constraints.

3.3.2 Solving VRPs by Savings with Look-Ahead. When the cluster partitioning is done, the constructed tours are discarded and each VRP is solved from scratch. In each iteration of the savings method, two routes with the largest savings value are connected. A saving value is the negation of total distance change when connecting two points to a route, as explained below [2]. Consider two customers y and z in a feasible solution with depot 0, where links $(y-1, y, y+1)$ and $(z-1, z, z+1)$ are part of their routes, respectively. The operation is linking (y, z) into one route, which leads to severing of the two links $(y-1, y, y+1)$ and $(z-1, z, z+1)$. There are 4 ways to do so, and the distances saved by each of the decompositions are (1) $c_{y,y+1} - c_{0,y+1} + c_{z,z+1} - c_{0,z+1} - c_{y,z}$, (2) $c_{y-1,y} - c_{0,y-1} + c_{z,z+1} - c_{0,z+1} - c_{y,z}$, (3) $c_{y,y+1} - c_{0,y+1} + c_{z,z-1} - c_{0,z-1} - c_{y,z}$, (4) $c_{y-1,y} - c_{0,y-1} + c_{z,z-1} - c_{0,z-1} - c_{y,z}$. The consideration is whether to cut $(y-1, y)$ or $(y, y+1)$, and whether to cut

$(z-1, z)$ or $(z, z+1)$. The change with the maximum saving value is selected, if it produces feasible routes for the capacity constraints.

The look-ahead procedure constructs a search tree, looking several stages into the future for the alternative choices. The search tree is configured such that each node at level i has a_i child nodes, for $i \geq 3, a_i = max\{1, a_2 - (i-2)^r\}$ for some r. The parameters are set as $a_1 = 20, a_2 = 1, r = 0$ and $depth = 50$. That is, the root expands the top 20 moves with the best savings, and each node at deeper levels only expands the top 1 move with the best saving value. As a result, a move is selected such that the corresponding leaf node has the best total savings.

After VRPs are solved, 2-opt and 3-opt are used to improve individual routes, and an inter-route swap operator, similar to the one used in (3.2.2).

LAR uses relatively simple operators, and it's easy to understand. The use of search tree provides a better approach to search the solution space. It would take a lot of effort the adjust the parameters for the search tree, if they are not already provided. Luckily there were already well-tuned based on the tradeoff between solution qualtiy and running time. The benchmark shows that LAR can provide solutions that are only slightly worse than the Tabu Search method, using only 0.5% of the running time on average [1].

3.4 Ant Colony Optimization

Ant colony optimization (ACO) [12, 13] simulates food-seeking behaviours of ant colonies in nature. Each ant simulates a vechile, and constructs its route by repeatedly selecting the next customer to visit, based on probablities explained below. The customers who have been already visited or violte the capacity constraints are stored in a tabu list.

3.4.1 Adding Virtual Depot to MDVRP. To begin with, a virtual central depot, 0, is added to the graph such that $c_{0,i} = \text{inf}$ for each customer i, and $c_{0,j} = 0$ for each actual depot j. Note that this algorithm no longer assumes all points are placed on a 2D plane. Each vechile starts its route from the virtual central depot, going though one actual depot to customers and returns through the same actual depot. This formulation of problem is called V-MDVRP [12].

3.4.2 Solution Generation. In ACO for VRP, the ant selects the next customer based on probabilities taking into account for both the visibility and the pheromone information on an edge. To selection of the next customer j the ant uses the un-normalized probability weight function:

$$p(i,j) = \begin{cases} (\tau(i,j))^{\alpha} \times (\eta(i,j))^{\beta} & \text{if } j \notin \text{tabu} \\ 0 & \text{otherwise} \end{cases}$$

where $p(i,j)$ is the probability weight to combine customers i and j on the route; $\tau(i,j) =$ the pheromone concentration of edge (i,j), which can tell how good this combination was in past iterations; $\eta(i,j) = 1/c_{i,j} =$ the visibility of edge (i,j). α and β are relative influence factors.

3.4.3 Genetic Mutation Operations. ACO was implemented in parallel environment as a genetic algorithm, with a population of chromosomes (multiple solution states). Therefore it is useful to define some mutation operation to maintain the genetic diversity. The first operation is selecting a route and replacing its depot with another depot. The second operation is removing a customer from its route and inserting it to another. If inserting a customer to another route leads to infeasibility, another route is added to fix the capacity violation. Also, 2-opt exchanges are used to optimized the new route induced by the second operation.

3.4.4 Updating Pheromone Information. The following pheromone updating equation is used:

$$\tau(i,j) = \rho \times \tau(i,j) + \Delta\tau(i,j)$$
$$\Delta\tau(i,j) = \sum_h \sum_k \Delta\tau_k^h(i,j)$$

where $\Delta\tau_k^h(i,j)$ = the pheromone increments on route k through the actual depot h on edge (i,j); $0 < \rho < 1$ is the speed of evaporation.

The ant-weight strategy [13] is used to assign the increased pheromone:

$$\Delta\tau_k^h(i,j) = \begin{cases} \frac{Q}{L} \times \frac{L-L^h}{(m-1)\times L} \times \frac{L^h - L_k^h}{(n^h-1)\times L^h} & \text{if edge } (i,j) \text{ on route } k \text{ through depot } h \\ 0 & \text{otherwise} \end{cases}$$

where, Q = a constant; L = total length of all routes in the solution; L^h = total length of routes through depot h; L_k^h = the length of route k through depot h; n^h = number of routes through depot h; m = number of actual depots. Note that $L = \sum_h L^h$, $L^h = \sum_k L_k^h$. Also, $\frac{Q}{L}$ is the total pheromone increments on all edges; $\frac{L-L^h}{(m-1)\times L}$ is the shared percentage increments on routes through depot h; and the increment on edge (i,j) is propotional to $\frac{L^h - L_k^h}{(n^h-1)\times L^h}$, which is the share of route k though depot h.

Most effort of ACO algorithm is spent on simulating the ant colony behaviours to construct the solution and updating the probability weights. Extra diversifiction comes from relatively simple mutation operators. Since the moves are probablistic, multiple solution states can be explored in parallel environment, with little information exchange or overlapping.

3.5 Coevolutionary Algorithm

This coevolutionary algorithm [3] published in year 2015 mainly uses genetic strategies. The MDVRP is decomposed into single-depot VRPs as well. Because each subproblem has its population evolving independently, this approach suits parallel environments very well.

3.5.1 Cluster Partitioning. Different from other algorithms described above, in the coevolutionary model each customer can be assigned to two depots. The first one is the closest depot and the second one is the closest depot to the closest customer neighbour, if differs from the first. This seems to make stronger ties between individual clusters and the complete solution, but additional approaches will needed to repair the duplicate customers when combining individual VRP solutions.

3.5.2 Cluster Polulation Initialization. After the partitioning, each cluster still represents a single-depot VRP, to be solved by an evolutionary algorithm. In the first stage, a semi-greedy strategy is used to generate a gaint route around the depot, ignoring the capacity constraints. The first customer is randomly selected, and the remaining customers are sorted by their distance from the previous customer. The next customer is selected at random from a list of α best customers. This process repeats until the gaint route is complete.

3.5.3 Population (Cluster) Evolution. The gaint route is considered as a chromosome, with a sequence of genes (customers), but without actual route boundaries. Actually, given the customer sequence on the gaint route, individual feasible routes can be constructed by the Split algorithm [8], which yields optimal solution in $O(n^2)$ time and $O(n)$ space for n customers. A random parent is selected to generate certain number of offsprings, by applying a random number of mutations. A mutation is applied to the gaint route such that two random customers are swapped in the sequence. Then the Split algorithm is used to generate individual routes on the offspring, followed

by a local search procedure to improve the routes. 9 operators are used, similar to 2-opt and 2-route exchanges described in 3.2.2. The local search stops when no improvement can be found. Once all offspring are created, parents and offspring are ranked by their local fitness, which is basically total length of routes, but can introduce penalty for an extra number of routes, if the depot has only fixed number of vehicles. Then the next population is selected based on the rank.

3.5.4 Complete Solutions. In order to produce complete solutions from a population of partial solutions, one population p is given such that each individual in p is combined with the best individuals from other populations. Next, a repair procedure is applied to remove duplicate customers, based on the least additional cost. The complete solution is also improved by the same local search procedure, where moving a customer to a different depot is also allowed.

A certain number of best MDVRP solutions so far are maintained as the Elite Group (EG). The best solution among them is called the guiding solution. The EG module attempts to progressivly modify an initial solution in (EG) to the guiding solution. It is hoped that during the process, a better solution than the guilding solution will emerge.

4 DISCUSSION

4.1 Benchmark Results

It may be surprising that the MDVRP instances designed by Cordeau are so widely used by different researchers. The instances can be found at http://www.bernabe.dorronsoro.es/vrp/ . All of the five algorithms described above were tested on these instances according to their sources. The multi-terminal sweep algorithm was tested on problems 1-11 [5] and the other four algorithms were tested at least on problems 1-23 [1, 3, 11, 12]. It might be unfair to [5] since it was tested in the 1970s, and its benchmark was not re-run by later researchers for comparison purpose. As a result, in the benchmark of [11], the tabu search algorithm gave better solutions than [5] on all of problems 1-11, with a highest improvement (decrement in the total cost) of 13% on problem 9. Achieving such an improvement on the NP-hard MDVRP is never easy. Despite the dramatic improvement on computing power over the 20 years, tabu search has been considered as a generally better approach than the sweep algorithm, and has been widely used by later researchers as a benchmark comparison.

The fast look-ahead algorithm [1] was benchmarked 10 years after the tabu search method. This newer approach, gave solutions that are 0.82% inferior when comparing with the tabu search method, but only spent 0.4% of the running time on average.

The parallized ant colony algorithm provided slightly better solutions when compared with the re-run benchmark of tabu search method, with a highest improvement of 3.2% on problem 21 [12].

Compared with the ant colony approach, the coevolutionary algorithm provided better solutions (no more than 1% better) on 7 of the 23 problems [3]. I cannot say it provides a significant improvement.

4.2 Which Technique to Use?

The fast look-ahead algorithm [1] is probably the best choice to get good solutions in a short time, which is also not difficult to implement. Among the five described algorithms, it may be the only

choice in case of real-time analysis because of its fast speed.

If you are very eager to get a even slightly better solution, despite of computing resource, you should probably pick one of the genetic algorithms [3, 12]. It is not hard to understand their concepts, but they can be hard to implement, due to the parallel programming requirement. They also expect more computing resources, to store and process population of solution states, instead of just one. Among the two, the parallel ant colony method is slightly better than the coevolutionary method, due to comparable results and simpler program structure.

4.3 Future Research

It would be useful to conduct some real-world case study, to construct solutions of different VRP variants, with more complex constraints. After all, working with only classical instances has its limit. Although these MDVRP algorithms give solutions with similar quality, it is possible that they will show more variance when handling larger cases. After all, just looking at the provided benchmarks, it seems that newer algorithms are not much better than the old ones.

Another point of interest would be the combination of different parts of these techniques. Combining different approches of clustering, route initialization, search operators, and solution merging can make various "new" algorithms. It would be interesting to compare the performance of each sub-procedure in detail, besides the overall benchmarks. A good example is that [11] benchmarked the 3 stages (fast improvement, intensification and diversification) separately, and showed that even only using the fast improvement stage, the tabu search could generate pretty good results.

REFERENCES

[1] Juergen Branke, Christian Schmidt, and Markus Withopf. 2007. A fast look-ahead heuristic for the multi-depot vehicle routing problem. *Wirtschaftinformatik Proceedings 2007* (2007), 80.

[2] Geoff Clarke and John W Wright. 1964. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research* 12, 4 (1964), 568–581.

[3] Fernando Bernardes de Oliveira, Rasul Enayatifar, Hossein Javedani Sadaei, Frederico Gadelha Guimarães, and Jean-Yves Potvin. 2016. A cooperative coevolutionary algorithm for the Multi-Depot Vehicle Routing Problem. *Expert Systems with Applications* 43 (2016), 117–130.

[4] Marshall L Fisher. 1994. Optimal solution of vehicle routing problems using minimum k-trees. *Operations research* 42, 4 (1994), 626–642.

[5] Billy E Gillett and Jerry G Johnson. 1976. Multi-terminal vehicle-dispatch algorithm. *Omega* 4, 6 (1976), 711–718.

[6] Billy E Gillett and Leland R Miller. 1974. A heuristic algorithm for the vehicle-dispatch problem. *Operations research* 22, 2 (1974), 340–349.

[7] Gilbert Laporte, Yves Nobert, and Danielle Arpin. 1984. *Optimal solutions to capacitated multidepot vehicle routing problems.* Université de Montréal, Centre de recherche sur les transports.

[8] Christian Prins. 2004. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research* 31, 12 (2004), 1985–2002.

[9] Jacques Renaud, Fayez F Boctor, and Gilbert Laporte. 1996. A fast composite heuristic for the symmetric traveling salesman problem. *INFORMS Journal on computing* 8, 2 (1996), 134–143.

[10] Jacques Renaud, Fayez F Boctor, and Gilbert Laporte. 1996. An improved petal heuristic for the vehicle routeing problem. *Journal of the operational Research Society* 47, 2 (1996), 329–336.

[11] Jacques Renaud, Gilbert Laporte, and Fayez F Boctor. 1996. A tabu search heuristic for the multi-depot vehicle routing problem. *Computers & Operations Research* 23, 3 (1996), 229–235.

[12] Bin Yu, ZZ Yang, and JX Xie. 2011. A parallel improved ant colony optimization for multi-depot vehicle routing problem. *Journal of the Operational Research Society* 62, 1 (2011), 183–188.

[13] Bin Yu, Zhong-Zhen Yang, and Baozhen Yao. 2009. An improved ant colony optimization for vehicle routing problem. *European journal of operational research* 196, 1 (2009), 171–176.