

# A Simulated Annealing Algorithm for The Capacitated Vehicle Routing Problem

H. Harmanani, D. Azar, N. Helal  
Department of Computer Science & Mathematics  
Lebanese American University  
Byblos, 1401 2010, Lebanon

W. Keirouz  
Department of Computer Science  
American University of Beirut  
Beirut, 1107 2020, Lebanon

## Abstract

The *Capacitated Vehicle Routing Problem* (CVRP) is a combinatorial optimization problem where a fleet of delivery vehicles must service known customer demands from a common depot at a minimum transit cost without exceeding the capacity constraint of each vehicle. In this paper, we present a meta-heuristic approach for solving the CVRP based on simulated annealing. The algorithm uses a combination of *random* and *deterministic operators* that are based on problem knowledge information. Experimental results are presented and favorable comparisons are reported.

## 1 Introduction

The vehicle routing problem (VRP) is a generalization of the traveling salesman problem, and was initially proposed by Dantzig et al. [13] in order to find the optimum routing of a fleet of gasoline delivery trucks between a bulk terminal and a large number of service stations supplied by the terminal. The *Capacitated VRP* (CVRP) is a variation of the problem where all the customers correspond to deliveries and the demands are deterministic, known in advance, and may not be split. The vehicles are identical and based at a single central depot, and only the capacity restrictions for the vehicles are imposed. The objective is to minimize a weighted function of the number of routes and their travel time while serving all customers [30]. Other variations of the problem add pickup and delivery constraints (VRPPD) and time windows constraints (VRPTW).

The vehicle routing problem is an important combinatorial optimization problem that has various real-life applications. The problem has been shown to be related to practical problems such as solid waste collection, street cleaning, bus routing, dial-a-ride systems, routing of maintenance units, transports for handicapped, and modern telecommunication networks.

The CVRP is known to be NP-hard (in the strong sense), and generalizes the well known *Traveling Sales-*

*man Problem (TSP)*, calling for the determination of a minimum-cost simple circuit visiting all the vertices of a graph, and arising when there is one vehicle only with the capacity  $C$  not smaller than the demand at each vertex [30]. Therefore no exact algorithm has been shown to consistently solve CVRP instances with more than 25 customers [11].

A variety of optimization approaches have been applied to the CVRP [6, 18, 19, 30]. Clarke et al. [12] proposed a heuristic for the VRP based on the notion of savings where a distance saving is generated if two routes can be feasibly merged into a single route. Baldacci et al. [5] proposed a two-commodity network flow formulation by deriving a new lower bound from linear programming LP relaxation. The algorithm is based on the extension to CVRP of the two-commodity flow formulation for the TSP. Other researchers have proposed optimization approaches that are based on the use of graph theoretical relaxations such as b-matchings [22], K-trees [14], dynamic programming [16], and set partitioning and column generation [1]. Branch-and-cut algorithms based on the “two-index formulation of the CVRP” have been proposed as well [2, 3, 8, 30, 24, 26]. Valid linear inequalities are used as cutting planes to strengthen a linear programming relaxation at each node of a branch-and-bound tree. Other solutions that are based on meta-heuristics have been proposed as well such as simulated annealing algorithms [23], tabu search [9, 15, 29], genetic algorithms [4, 7, 25], honey bees mating optimization [20], and ant colony algorithm [21, 27, 28]. A comprehensive survey of the problem is provided in [30].

In this paper, we present an effective simulated annealing algorithm for solving the capacitated vehicle routing problem (CVRP). The proposed algorithm combines the strong global search ability of simulated annealing with problem knowledge information in order to provide effective and competitive solutions. The remainder of the paper is organized as follows. Section 2 describes the CVRP problem while section 3 introduces simulated annealing. Section 4 formulates the *annealing CVRP* and describes the *formulation, cool-*

ing schedule, the neighborhood function, and the cost function. The *CVRP annealing algorithm* is described in section 5 while *experimental results* are presented in section 6. We *conclude* with remarks in section 7.

## 2 Problem description

The CVRP involves determining a set of routes, starting and ending at the depot  $v_0$ , that cover a set of customers. Each customer has a given demand and is visited exactly once by exactly one vehicle. All vehicles have the same capacity and carry a single kind of commodity. No vehicle can service more customers than its capacity  $C$  permits. The objective is to minimize the total distance traveled or the number of vehicles used, or a combination of both. Thus, the CVRP is reduced to partitioning the graph into  $m$  simple circuits where each circuit corresponds to a vehicle route with a minimum cost such that: (1) each circuit includes the depot vertex; (2) each vertex is visited by exactly one circuit; (3) the sum of the demands of the vertices by a circuit does not exceed the vehicle capacity  $C$ .

Formally, the CVRP can be defined as follows: Given a complete undirected graph  $G = (V, E)$  where  $V = v_0, v_1, \dots, v_n$  is a vertex set and  $E = (v_i, v_j) / v_i, v_j \in V, i < j$  is an edge set. Vertex  $v_0$  denotes the depot, and it is from where  $m$  identical vehicles of capacity  $C$  must serve all the cities or customers, represented by the set of  $n$  vertices  $v_1, \dots, v_n$ . Each edge is associated with a non-negative cost, distance or travel time  $c_{ij}$ , between customers  $v_i$  and  $v_j$ . Each customer  $v_i$  has non-negative demand of goods  $q_i$  and drop time  $\delta_i$  (time needed to unload all goods). Let  $V_1, \dots, V_m$  be a partition of  $V$ , a route  $R_i$  is a permutation of the customers in  $V_i$  specifying the order of visiting them, starting and finishing at the depot  $v_0$ . The cost of a given route  $R_i = v_{i0}, v_{i1}, \dots, v_{ik+1}$ , where  $v_{ij} \in V$  and  $v_{i0} = v_{ik+1} = 0$  (0 denotes the depot), is given by:

$$Cost(R_i) = \sum_{j=0}^k c_{j,j+1} + \sum_{j=0}^k \delta_j \quad (1)$$

and the cost of the problem solution (S) is:

$$F(S)_{CVRP} = \sum_{i=1}^m Cost(R_i). \quad (2)$$

## 3 Simulated Annealing

Simulated annealing is a global stochastic method that is used to generate approximate solutions to very

large combinatorial problems and was first introduced by Kirkpatrick et al. [17]. The annealing algorithm begins with an initial feasible configuration and proceeds to generate a neighboring solution by perturbing the current solution. If the cost of the neighboring solution is less than that of the current solution, the neighboring solution is accepted; otherwise, it is accepted or rejected with probability  $p = e^{-\frac{\Delta C}{T}}$ . The probability of accepting inferior solutions is a function of the *temperature*,  $T$ , and the change in cost between the neighboring solution and the current solution,  $\Delta C$ . The temperature is decreased during the optimization process and thus the probability of accepting a worse solution decreases as well. The set of parameters controlling the initial temperature, stopping criterion, temperature decrement between successive stages, and the number of iterations for each temperature is called the *cooling schedule*. Typically, at the beginning of the algorithm, the temperature  $T$  is large and an inferior solution has a high probability of being accepted. During this period, the algorithm acts as a random search to find a promising region in the solution space. As the optimization process progresses, the temperature decreases and there is a lower probability of accepting an inferior solution. The algorithm behaves like a down hill algorithm for finding the local optimum of the current region.

## 4 Problem Formulation

The proposed method starts with a graph of demands and generates, through a sequence of transformations, a set of sub-optimal routes. The key elements in implementing the annealing CVRP algorithm are: 1) the definition of the initial configuration, 2) the definition of a neighborhood on the configuration space and perturbation operators exploring it; 3) the choice of the cost function; and 4) a cooling schedule. In what follows, we describe our annealing algorithm with reference to Figure 1.

### 4.1 Configuration Representation

In order to solve the CVRP problem, we propose the configuration shown in Figure 1(a). The representation is based on a vector where each cell  $m \in 1, 2, 3, \dots, n$  corresponds to a customer and cell 0 denotes the depot. Each route starts and ends at the depot. For example, Figure 1(b) encodes the following three routes:  $\langle v_0, v_1, v_3, v_7, v_0 \rangle$ ,  $\langle v_0, v_2, v_4, v_0 \rangle$  and  $\langle v_0, v_6, v_5, v_8, v_0 \rangle$ .

In order to speed-up the operation, we maintain hash tables that represent the routes gener-

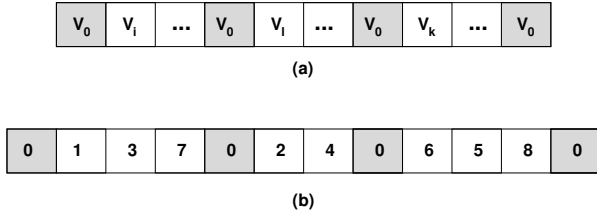


Figure 1: (a) Configuration Representation, (b) Sample Configuration

ated as well as the corresponding capacity. Thus, in Figure 1, the following hash indexes are maintained for the routes:  $(r_0, 0, 1, 3, 7, 0)$ ,  $(r_1, 0, 2, 4, 0)$ , and  $(r_2, 0, 6, 5, 8, 0)$ . Similarly, the following hash indexes are maintained for the capacity of each route:  $(r_0, \text{total demand in } r_0)$ ,  $(r_1, \text{total demand in } r_1)$ , and  $(r_2, \text{total demand in } r_2)$ .

## 4.2 Initial Configuration

The initial solution is generated deterministically using a greedy algorithm that iterates over the list of cities and constructs initial non-optimal feasible routes based on a first-fit approach. The algorithm, shown in Figure 2, proceeds as follows. If there are  $n$  customers  $1, 2, 3, \dots, n$ , the algorithm assigns the first customer in the list to a new route as long as its demand doesn't violate the capacity constraint of a vehicle and sets the status of the node to be visited. The algorithm proceeds to the next customer in the list. If it encounters a customer demand that violates the capacity constraint, the customer is skipped and the following customer in the list is processed. If the route capacity is exceeded, then a new route is allocated and the algorithm repeats until all customers have been assigned to routes.

```

InitialSolution()
{
   $i \leftarrow 0$ 
  do {
    Allocate a new route  $r_i$ .
     $j \leftarrow 0$ 
     $\forall v_j$  {
      if  $(v_j.\text{visited} == \text{false} \ \&\& \ C(r_j) < Q)$  then
         $r_i \leftarrow r_i \cup v_i$ 
         $r_i.\text{visited} \leftarrow \text{true}$ 
      }
    }
     $j++$ 
  }
}

```

Figure 2: Initial Solution

## 4.3 Neighborhood Transformation

The annealing algorithm uses two transformations in order to explore the design space. In what follows, we present both transformations.

### 4.3.1 Move

The *move* transformation finds *five* pairs of customers  $< v_i, v_{i+1} >$  that have the shortest distances closest to each other including the depot. This is done by computing all distances between each pair of customers on all generated routes, including the distances to the depot. The transformation next selects five random customers that *exclude* the depot and the customers at positions  $v_{i+1}$ . The random customers are removed from their routes, and deterministically inserted into random routes. The transformation selects a random route and inserts the random customers in the routes based on the capacity constraint. Thus, for every random customer, a random route is selected and if the customer demand does not violate the route capacity it will be inserted in the new route.

### 4.3.2 Replace Highest Average

The *replace highest average* transformation calculates the average distance of every pair of customers in the graph. Thus, for each vertex  $v_i$  in a route, the method computes  $d_i = \frac{d_{i-1,i} + d_{i,i+1}}{2}$  and selects the five vertices with the largest average distances and removes them from their routes. The transformation picks next five random routes and inserts the five selected customers in the route with the resulting minimum cost.

## 4.4 Cost Function

The objective of the algorithm is to minimize the cost of all routes. Thus, the objective is to minimize the following problem cost function:

$$f_S = \sum_{i=1}^m \text{Cost}(R_i). \quad (3)$$

## 4.5 Cooling Schedule

The cooling schedule is the set of parameters controlling the *initial temperature*, the *stopping criterion*, the *temperature decrement* between successive stages, and the *number of iterations* for each temperature. The cooling schedule was empirically determined. Thus, the initial temperature,  $T_{init}$ , was set to 5000 while the temperature reduction multiplier,  $\alpha$ , was set to 0.99. The number of iterations,  $M$ , was determined to be 5 while the iteration multiplier,  $\beta$  was

set to 1.05. The algorithm stops when the temperature,  $T_f$ , is below 0.001.

## 5 CVRP Annealing Algorithm

Each configuration represents an intermediate route that has a different cost. During every annealing iteration, the neighborhood of the configuration is explored. The algorithm must ensure the following:

1. Each route originates and terminates at the depot;
2. The total demand for each route is within the capacity limit;
3. The maximum traveling distance is not surpassed;
4. The number of routes generated does not exceed the number of vehicles.

The algorithm, shown in Figure 3, starts by selecting an initial configuration and then a sequence of iterations is performed. In each iteration a new configuration is generated in the neighborhood of the original configuration by replacing vertices that have the largest distances to other vertices as well as by moving the vertices with the highest average distance to their neighbors. The *move* transformation is applied 80% of the time while the *replace highest* transformation is applied in every iteration. The solution is always feasible as the neighborhood transformations use a constructive approach that generates feasible routes. The variation in the cost functions,  $\Delta_C$ , is computed and if negative then the transition from  $C_i$  to  $C_{i+1}$  is accepted. If the cost function increases, the transition is accepted with a probability based on the Boltzmann distribution. The temperature is gradually decreased from a sufficiently high starting value,  $T_{init} = 5000$ , where almost every proposed transition, positive or negative, is accepted to a freezing temperature,  $T_f = 0.001$ , where no further changes occur.

## 6 Experimental Results

The proposed algorithm was implemented using Java on a Pentium Core 2 duo 1.80 GHZ with 1 GB of RAM, and tested on various instances from series A and E that are available at [www.branchandcut.org](http://www.branchandcut.org).

### 6.1 A Benchmarks

The first set of benchmarks that we have attempted are the A instances from Augerat [3]. The

```

Annealing_CVRP()
{
     $S_0$  = Initial solution
     $\alpha = 0.99$  //Temperature reduction multiplier
     $\beta = 1.05$  // Iteration multiplier
     $M_0 = 5$  //Time until next parameter update
    BestS = Best solution
     $T = 5000$ 
    CurrentS =  $S_0$ 
    CurrentCost = Cost(CurrentS)
    BestCost = Cost(BestS)
    Time = 0
    do {
         $M = M_0$ 
        do {
            NewS = Neighbor(CurrentS);
            NewCost = Cost(NewS)
             $\Delta_{Cost} = NewCost - CurrentCost$ 
            If ( $\Delta_{Cost} < 0$ )
                CurrentS = NewS
                CurrentCost = Cost(CurrentS);
                If ( $NewCost < BestCost$ ) then
                    BestS = NewS
                    BestCost = Cost(BestS)
            else if (Random <  $e^{-\frac{\Delta_{Cost}}{T}}$ ) then
                CurrentS = NewS
                CurrentCost = Cost(CurrentS);
                 $M = M - 1$ 
        } while ( $M \geq 0$ )
        Time = Time +  $M_0$ ;
         $T = \alpha * T$ ;
         $M_0 = \beta * M_0$ ;
    } while (Time > MaxTime and  $T > 0.001$ );
    Return(BestS);
}

```

Figure 3: Annealing CVRP Algorithm

first field in the name refers to the problem type where A refers to the *asymmetric capacitated vehicle routing problem* (ACVRP) instances. The second field is a three-digit integer that denotes the number of vertices of the problem graph, including the depot vertex. The third field is a one-digit integer that denotes the number of available vehicles. All examples were proposed by Augerat [3]. For example, A-n32-k5 identifies the classical 32-customers Euclidean instance with 5 available vehicles. All examples are Euclidean, with integer edge costs following the TSPLIB standard. For the instances in the class A, both customer locations and demands are random. The algorithm solved all problems in under 15 minutes. Each benchmark was solved for 10 times and the best, worst and average results are reported in Table 2. All results were within 5% of the reported optimum or best answer with one case where our algorithm found a new optimum, E-n23-k3.

### 6.2 E Benchmarks

We have attempted three large benchmarks that are part of the E benchmark series. The first field in the name is the problem type where E refers to Euclidean single-depot complex vehicle routing problem instances (SCVRP). The second field is a three-digit integer that denotes the number of vertices of the prob-

Benchmark	Best Known Optimal	Simulated Annealing			Std Deviation $\sigma$	Difference from Optimal
		Best	Worst	Average		
E051-05e	524.61	555.14	573.09	562.72	6.73	5.82%
E076-10e	835.26	910.32	974.55	940.06	19.89	7.92%
E101-08e	826.14	1051.34	1091.17	1073.94	11.9	26.72%

Table 1: Results for the E instances

lem graph. The third field is a two-digit integer that denotes the number of available vehicles. All examples were proposed by Christofides and Eilon [10]. For example, E051-05e identifies the classical 50-customers Euclidean instance with 5 available vehicles. The examples are relatively large and include 50, 75, and 100 cities. Each benchmark was attempted for 10 times. The running time for the algorithm was 13 minutes in all cases. We compare our results in Table 1 to the reported optimal routes. It can be shown that the results of the first benchmark are very close to the reported optimal and differs by about 5% for the 50 cities and 8% for the 75 cities. The results were 26.72% from the optimal solution in the case of the 100 cities.

## 7 Conclusion

We have presented a meta-heuristic solution for the *Capacitated Vehicle Routing Problem*. The problem is solved based on simulated annealing using a combination of *random* and *deterministic operators* that are based on problem knowledge information. The algorithm was implemented and various benchmarks were attempted. The reported results are promising and future directions will focus on parallelization in order to further improve the results.

## References

- [1] Agarwal, Y., Mathur, K., Salkin, H.M.: “A Set-Partitioning-Based Exact Algorithm for the Vehicle Routing Problem, *Networks*, Vol. 19, pp. 731–749, 1989.
- [2] J.R. Araque, G. Kudva, T.L. Morin, J.F. Pekny, “A Branch-and-Cut Algorithm for the Vehicle Routing Problem, *Ann. Oper. Res.*, Vol. 50, pp. 37–59, 1994.
- [3] P. Augerat, “Approche Polyédrale du Problème de Tournees de Véhicules,” *PhD Thesis, Institut National Polytechnique de Grenoble*, 1995.
- [4] B.M. Baker and M.A. Ayechew, “A Genetic Algorithm for the Vehicle Routing Problem,” *Computers & Oper. Research* Vol. 30, pp. 787–800, 2003.
- [5] R. Baldacci, E. Hadjiconstantinou, and A. Mingozzi, “An Exact Algorithm for the Capacitated Vehicle Routing Problem Based on a Two-Commodity Network Flow Formulation,” *Operations Research*, Vol. 52, pp. 723–738, 2004.
- [6] M.O. Ball, T.L. Magnanti, C.L. Monma, and G.L. Nemhauser, (eds.) *Network Routing: Handbooks on Operations Research and Management Science*, Vol. 8, 1995
- [7] J. Berger and M. Barkaoui, “A Hybrid Genetic Algorithm for the Capacitated Vehicle Routing Problem,” *Journal of Operation Research Society*, Vol. 54, pp. 1254–1262, 2003.
- [8] U. Blasum, W. Hochstättler, “Application of the Branch-and-Cut Method to the Vehicle Routing Problem,” *Technical Report, Universität zu Köln*, 2002
- [9] J. Brandao and R.A. Eglese, “A Deterministic Tabu Search Algorithm for the Capacitated Arc Routing Problem,” *Computers & Operations Research*, Vol. 35, pp. 1112–1126, 2008.
- [10] N. Christofides and S. Eilon, “An algorithm for the Vehicle Dispatching Problem,” *Operational Research Quarterly*, Vol. 20, pp. 309–318, 1969.
- [11] N. Christofides, A. Mingozzi, and P. Toth, “Exact Algorithms for the Vehicle Routing Problem Based on Spanning Tree and Shortest Path Relaxations,” *Mathematical Programming*, Vol. 20, pp. 255–282, 1981.
- [12] G. Clarke and J.V. Wright, “Scheduling of Vehicles From a Central Depot to a Number of Delivery Points,” *Operations Research*, Vol. 12, pp. 568–581, 1964.
- [13] G. B. Dantzig and J. H. Ramser, “The Truck Dispatching Problem,” *Management Science* Vol. 6, pp. 80–91, 1959.
- [14] M.L. Fisher, “Optimal Solution Of Vehicle Routing Problems Using Minimum K-Trees, *Operations Research*, Vol. 42, pp. 626–642, 1994.

Benchmark	Best Known Optimal	Simulated Annealing			Std Deviation $\sigma$	Difference from Optimal
		Best	Worst	Average		
E-n23-k3	569	568.5625	571.2862	569.3499	0.93	-0.08%
A-n32-k5	784	791.0789	845.704	818.0189	19.11	0.9%
A-n33-k5	661	685.577	704.602	696.9174	6.945	3.72%
A-n33-k6	742	748.08	767.53	756.3908	7.23	0.82%
A-n34-k5	778	802.614	819.388	812.5855	5.166	3.16%
A-n36-k5	799	837.477	883.809	862.2379	15.65	4.82%
A-n37-k5	669	697.943	745.564	725.473	15.58	4.33%
A-n37-k6	949	963.995	1000.84	981.1535	12.97	1.58%
A-n38-k5	730	733.945	773.824	763.19	11.76	0.54%
A-n39-k5	822	848.322	877.3145	865.427	9.75	3.2%
A-n39-k6	831	859.147	924.254	893.18	20.48	3.39%
A-n44-k6	937	951.802	1020.155	989.044	24.68	1.58%
A-n45-k6	944	982.829	1035.32	1010.898	14.55	4.11%
A-n45-k7	1146	1199.50	1313.74	1269.13	34.79	4.67%
A-n53-k7	1010	1044.91	1118.59	1087.58	22.51	3.46%
A-n54-k7	1167	1192.69	1275.43	1250.78	26.07	2.2%
A-n55-k7	1073	1144.85	1203.59	1175.15	21	6.7%
A-n60-k9	1408	1455.63	1567.34	1511.54	41	3.38%
A-n61-k9	1035	1064.76	1145.74	1101.70	23.63	2.88%
A-n62-k8	1290	1484.81	1633.13	1547.42	61.35	15.1%
A-n63-k9	1634	1698.63	1733.40	1715.08	12.23	3.96%

Table 2: Results for the A instances

- [15] A. Gendreau, A. Hertz, and G. Laporte, “A Tabu Search Heuristic for the Vehicle Routing Problem,” *Management Science*, Vol. 40, pp. 1276–1290, 1994.
- [16] E. Hadjiconstantinou, N. Christofides, A. Mingozzi, “A New Exact Algorithm For The Vehicle Routing Problem Based On Q-Paths And K-Shortest Paths Relaxations,” *Ann. Oper. Res.*, Vol. 61, pp. 21–43, 1996.
- [17] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by Simulated Annealing”, *Science*, Vol. 220, 671–680, 1983.
- [18] G. Laporte, “The Vehicle Routing Problem: An Overview of Exact and Approximate Algorithms,” *European Journal of Operations Research*, Vol. 59, pp. 345–358, 1992
- [19] G. Laporte, “Vehicle Routing,” *Annotated Bibliographies in Combinatorial Opt.*, Wiley, 1997
- [20] Y. Marinakis, M. Marinaki, and G. Dounias, “Honey Bees Mating Optimization Algorithm for the Vehicle Routing Problem,” *Studies in Computational Intelligence*, Vol. 129, pp. 139–148, 2008.
- [21] S. Mazzeo and I. Loiseau, “An Ant Colony Algorithm for the Capacitated Vehicle Routing,” *Electronic Notes in Discrete Math.*, Vol. 18, pp. 181–186, 2004.
- [22] D.L. Miller, “A Matching-Based Exact Algorithm For Capacitated Vehicle Routing Problems,” *ORSA J. Comp*, Vol. 7, pp. 1–9, 1995.
- [23] I. Osman, “Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem,” *Annals of Operations Research*, Vol. 41, pp. 421–451, 1993.
- [24] N.W. Padberg, G. Rinaldi, “A Branch-and-Cut Algorithm for the Resolution of Large-scale Symmetric Traveling Salesman Problems,” *SIAM Rev.*, Vol. 33, pp. 60–100, 1991.
- [25] C. Prins, “A Simple And Effective Evolutionary Algorithm For The Vehicle Routing Problem,” *Computers & Operations Research* Vol. 31, pp. 1985–2002, 2004.
- [26] T.K. Ralphs, L. Kopman, W.R. Pulleyblank, L.E. Trotter, “On the Capacitated Vehicle Routing Problem,” *Math. Program*, Vol. 94, pp. 343–359, 2003.
- [27] M. Reimann, M. Stummer, and K. Doerner, “A Savings Based Ant System for the Vehicle Routing Problem,” *In Proc. of the Genetic and Evolutionary Computation Conf.*, pp. 1317–1326, 2003.
- [28] M. Reimann, K. Doerner, and R. Hartl, “D-Ants: Savings Based Ants Divide and Conquer the Vehicle Routing Problem,” *Computers & Operations Research*, Vol. 31, pp. 563–591, 2004.
- [29] E.D. Taillard, “Parallel Iterative Search Methods for vehicle routing problems,” *Networks*, Vol. 23, pp. 661–672, 1993.
- [30] P. Toth and D. Vigo, *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, 2002