A Cooperative Coevolutionary Algorithm for the Multi-Depot Vehicle Routing Problem

Fernando Bernardes de Oliveira
Rasul Enayatifar
Hossein Javedani Sadaei
Frederico Gadelha Guimarães
Jean-Yves Potvin

February 2016

CIRRELT-2016-08

A Cooperative Coevolutionary Algorithm for the Multi-Depot Vehicle Routing Problem

Fernando Bernardes de Oliveira[1,2], Rasul Enayatifar[1], Hossein Javedani Sadaei[1], Frederico Gadelha Guimarães[3], Jean-Yves Potvin[4,*]

[1] Graduate Program in Electrical Engineering, Federal University of Minas gerais, Av. Antônio Carlos 6627, Belo Horizonte, MG 1270-901 Brazil

[2] Departamento de computao e Sistemas, Universidade Federal de Ouro Preto, UFOP, Joo Molevande, R. Diogo de Vasconcelos, 122 Pilar, Ouro Preso, MG, 35400-000 Brazil

[3] Department of Electrical Engineering, Universidade Federal de Minas Gerais, UFMG, Belo Horizonte, Av. Antônio Carlos, 6627 Pampulha, Belo Horizonte MG, 31270-901 Brazil

[4] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Computer Science and Operations Research, Université de Montréal, P.O. Box 6128, Station Centre-Ville, Montréal, Canada H3C 3J7

Abstract. The Multi-Depot Vehicle Routing Problem (MDVRP) is an important variant of the classical Vehicle Routing Problem (VRP), where the customers can be served from a number of depots. This paper introduces a cooperative coevolutionary algorithm to minimize the total route cost of the MDVRP. Coevolutionary algorithms are inspired by the simultaneous evolution process involving two or more species. In this approach, the problem is decomposed into smaller subproblems and individuals from different populations are combined to create a complete solution to the original problem. This paper presents a problem decomposition approach for the MDVRP in which each subproblem becomes a single depot VRP and evolves independently in its domain space. Customers are distributed among the depots based on their distance from the depots and their distance from their closest neighbor. A population is associated with each depot where the individuals represent partial solutions to the problem, that is, sets of routes over customers assigned to the corresponding depot. The fitness of a partial solution depends on its ability to cooperate with partial solutions from other populations to form a complete solution to the MDVRP. As the problem is decomposed and each part evolves separately, this approach is strongly suitable to parallel environments. Therefore, a parallel evolution strategy environment with a variable length genotype coupled with local search operators is proposed. A large number of experiments have been conducted to assess the performance of this approach. The results suggest that the proposed coevolutionary algorithm in a parallel environment is able to produce high-quality solutions to the MDVRP in low computational time.

Keywords: Multi-depot vehicle routing problem, cooperative coevolutionary algorithm, evolution strategies.

Acknowledgements. The authors would like to thank the following governmental agencies for their financial support. F.B. Oliveira: CAPES Foundation, Ministry of Education of Brazil, grant BEX 0295/14-0; R. Enayatifar and H. Javedani Sadaei: Brazilian Agency CAPES; F.G. Guimarães: National Council for Scientific and Technological Development, CNPq grant no. 312276/2013-3, Brazil; J.-Y. Potvin: Natural Sciences and Engineering Research Council of Canada (NSERC).

* Corresponding author: Jean-Yves.Potvin@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2016

1 Introduction

A vehicle routing problem (VRP) is a generic name for a large class of combinatorial optimization problems (Doerner and Schmid, 2010; Montoya-Torres et al., 2015). The goal is to find a set of routes for serving customers with a certain number of vehicles in a given environment. In the classical VRP, a problem instance is specified by a set of customers to be served with their corresponding locations and demands and other primary information such as distance between two costumers, distance between a customer and the depot, number of vehicles and vehicle capacity (Baldacci and Mingozzi, 2009). In a solution, each vehicle leaves the depot and executes a route over a certain number of customers before returning to the depot, while insuring that the total demand on the route does not exceed vehicle capacity. In some cases, a maximum route duration (or distance) constraint is enforced. The Multi-Depot Vehicle Routing Problem (MDVRP) is a variant of the classical VRP in which more than one depot is considered (Cordeau and Maischberger, 2012; Escobar et al., 2014; Subramanian et al., 2013; Vidal et al., 2012).

The number of studies on the MDVRP is rather limited when compared to the classical VRP. A survey of these studies, based on either exact methods or heuristics, can be found in Montoya-Torres et al. (2015). In recent years, evolutionary-based metaheuristics proved to be a popular approach to address this problem, as described in Section 3. But, in spite of this popularity, no coevolutionary algorithm has yet been proposed in the literature for the MDVRP. As the problem can be easily decomposed into a number of single-depot VRPs, with a population of partial solutions associated with each depot, a coevolutionary approach looks relevant. Each partial solution or individual in a population corresponds to vehicle routes defined over the subset of customers assigned to the corresponding depot. Although each population can evolve separately, this evolution is guided by the ability of each individual to form good complete solutions with individuals from the other populations. This is the problem-solving approach proposed in this work.

The remainder of this paper is organized as follows. First, some preliminaries about the MDVRP and coevolution are found in Section 2. Section 3 presents a literature review. Sections 4 and 5 describe the proposed methodology while Section 6 reports computational results. Future avenues for research are proposed in the conclusion in Section 7.

2 Preliminaries

In this section, some preliminary information about the mathematical formulation of the MDVRP and cooperative coevolutionary algorithms are presented.

2.1 Multi-Depot Vehicle Routing Problem Formulation

As mentioned earlier, the MDVRP is a variant of the classical VRP where more than one depot is considered (Montoya-Torres et al., 2015). Figure 1 shows a typical solution of this problem with two depots and two vehicle routes associated with each depot. Typically, the fleet of vehicles is limited and homogeneous (Cordeau and Maischberger, 2012; Escobar et al., 2014; Montoya-Torres et al., 2015; Subramanian et al., 2013; Vidal et al., 2012).

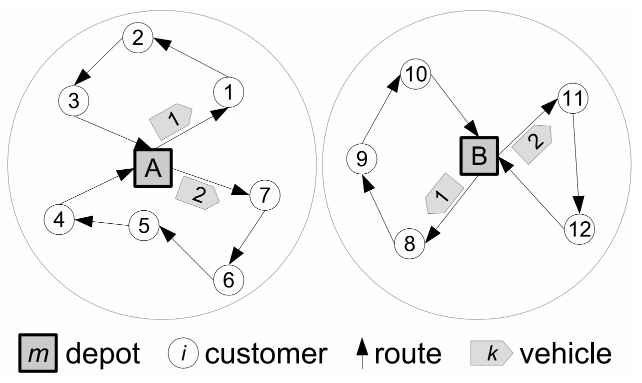


Figure 1: MDVRP solution.

Basically, a solution to this problem is a set of vehicle routes such that: (i) each vehicle route starts and ends at the same depot, (ii) each customer is served exactly once by one vehicle, (iii) the total demand on each route does not exceed vehicle capacity (iv) the maximum route time is satisfied and (v) the total cost is minimized (Montoya-Torres et al., 2015).

The MDVRP can be formalized as follows. Let $G = (V, A)$ be a complete graph, where V is the set of nodes and A is the set of arcs. The nodes are partitioned into two subsets: the customers to be served, $V_C = \{1, ..., N\}$, and the multiple depots $V_D = \{N+1, ..., N+M\}$, with $V_C \cup V_D = V$ and $V_C \cap V_D = \emptyset$. There is a non-negative cost c_{ij} associated with each arc $(i, j) \in A$. The demand of each customer is d_i (there is no demand at the depot nodes). There is also a fleet of K identical vehicles, each with capacity Q. The service time at each customer i is t_i while the maximum route duration time is set to T. A conversion factor w_{ij} might be needed to transform the cost c_{ij} into time units. In this work, however, the cost is the same as the time and distance units, so $w_{ij} = 1$.

In the mathematical formulation that follows, binary variables x_{ijk} are equal to 1 when vehicle k visits node j immediately after node i. Auxiliary variables y_i are also used in the subtour elimination constraints.

$$\text{Minimize} \sum_{i=1}^{N+M} \sum_{j=1}^{N+M} \sum_{k=1}^{K} c_{ij} x_{ijk} \ , \tag{1}$$

subject to:

$$\sum_{i=1}^{N+M}\sum_{k=1}^{K} x_{ijk} = 1 \quad (j = 1, ..., N); \tag{2}$$

$$\sum_{j=1}^{N+M}\sum_{k=1}^{K} x_{ijk} = 1 \quad (i = 1, ..., N); \tag{3}$$

$$\sum_{i=1}^{N+M} x_{ihk} - \sum_{j=1}^{N+M} x_{hjk} = 0 \quad (k = 1, ..., K; \quad h = 1, ..., N + M); \tag{4}$$

$$\sum_{i=1}^{N+M}\sum_{j=1}^{N+M} d_i x_{ijk} \leq Q \quad (k = 1, ..., K); \tag{5}$$

$$\sum_{i=1}^{N+M}\sum_{j=1}^{N+M} (c_{ij} w_{ij} + t_i)\, x_{ijk} \leq T \quad (k = 1, ..., K); \tag{6}$$

$$\sum_{i=N+1}^{N+M}\sum_{j=1}^{N} x_{ijk} \leq 1 \quad (k = 1, ..., K); \tag{7}$$

$$\sum_{j=N+1}^{N+M}\sum_{i=1}^{N} x_{ijk} \leq 1 \quad (k = 1, ..., K); \tag{8}$$

$$y_i - y_j + (M + N)x_{ijk} \leq N + M - 1; \text{ for } 1 \leq i \neq j \leq N \text{ and } 1 \leq k \leq K; \tag{9}$$

$$x_{ijk} \in \{0, 1\} \quad \forall \;\; i, j, k; \tag{10}$$

$$y_i \in \{0, 1\} \quad \forall \;\; i; \tag{11}$$

The objective (1) minimizes the total cost. Constraints (2) and (3) guarantee that each customer is served by exactly one vehicle. Flow conservation is guaranteed through constraint (4). Vehicle capacity and route duration constraints are found in (5) and (6), respectively. Constraints (7) and (8) check vehicle availability. Subtour elimination constraints are in (9). Finally, (10) and (11) define x and y as binary variables.

In the original formulation of the MDVRP, a fixed number of vehicles is allocated to each depot. In our work, though, the search is allowed to consider a larger number of vehicles (at a penalty cost in the objective). This is discussed in Section 5.

2.2 Coevolutionary Algorithms

Coevolutionary Algorithms (CA) is a class of evolutionary algorithms inspired by the simultaneous evolution process involving two or more species. Recently, various engineering problems have been solved with this approach (Blecic et al., 2014; Chen et al., 2014; Ladjici and Boudour, 2011; Ladjici et al., 2014; Wang and Chen, 2013; Wang et al., 2014). Coevolutionary algorithms are categorized into two groups depending on the type of interaction among the species, which can be either competitive or cooperative. *Competitive coevolution* can be viewed as an *arms race*, that is, individuals in the populations compete among themselves. One group attempts to take advantage over another, which responds with an adaptive strategy to recover the advantage (Katada and Handa, 2010). A biological example is the *predator-prey* competitive coevolution, in which the evolution of one population affects the evolution of the other. This approach has been used in virtual and simulated evolution (Ebner, 2006) and also for evolving agent behaviors or artificial players for games (Engelbrecht, 2007). In *Cooperative coevolution*, the interaction among species is beneficial for all or, at least, does not cause any damage. The species may live together in the same area and one population needs the other ones to survive and evolve. Typically, species cooperate to attain a global benefit. This is the case in *symbiosis*, for example (Engelbrecht, 2007).

In a cooperative coevolutionary algorithm, each population represents a part of a complex decomposable problem (Engelbrecht, 2007). Accordingly, the true fitness of an individual can only be obtained from its interaction with other individuals from the same or other populations. Each individual receives a reward or punishment, being rewarded when it interacts well with the others while getting a punishment otherwise. A cooperative coevolutionary algorithm is considered in this work because the problem can be decomposed into smaller subproblems, each one evolving in parallel. Partial solutions for those subproblems cooperate to create a complete solution for the MDVRP. In this situation, a competitive strategy would not be suitable because a complete solution is obtained from information gathered from all subproblems and there is no competition among these subproblems. Figure 2 depicts a general cooperative coevolutionary algorithm with decomposition. A complex problem is first decomposed into smaller subproblems. Each population is evolved on its subproblem and, after a number of generations, individuals from these populations are combined to create complete solutions to the original problem. Through this process, it is possible to compute the fitness of these complete solutions. Some feedback information is then returned to each population, such as the best solution found so far, any required updates to the individuals in the population, etc.

Coevolutionary Algorithms can be distinguished from traditional evolutionary algorithms by their evaluation process (Ficici, 2004). As mentioned above, individuals can only be evaluated through interaction with other individuals from the same or different populations.

Fitness Sampling (Engelbrecht, 2007) or *Fitness Assessment* (Luke, 2013) defines how individuals are combined for the purpose of fitness evaluation. These

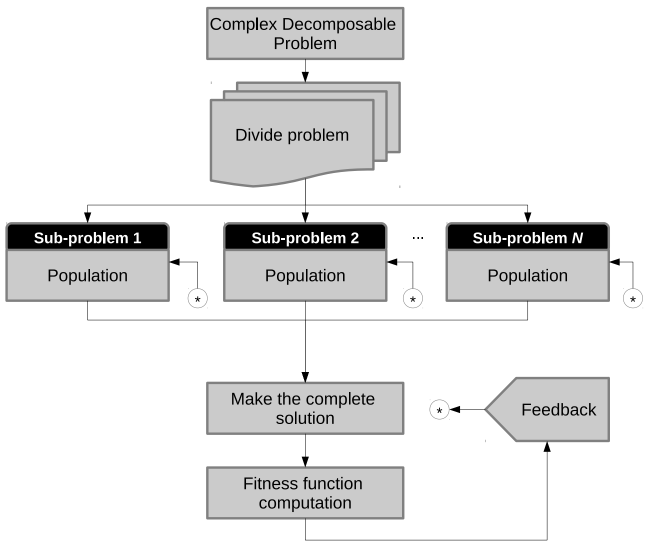


Figure 2: Cooperative coevolutionary algorithm with problem decomposition.

methods are the followings (Engelbrecht, 2007; Luke, 2013):

a) **All versus All**: All possible combinations of individuals from all populations are considered. This method is very expensive and can be appropriate for populations with only a few individuals.

b) **Random**: Individuals are randomly selected from the coevolving populations. This method is less expensive than the previous one and the number of evaluations is typically a parameter of the method.

c) **All versus best**: All individuals of one population are combined with the best individuals from other populations. This is repeated for each population.

d) **Tournament sampling**: This process consists of selecting individuals from each population based on their fitness. Then, a tournament is performed among all selected individuals to determine a winner. This is typically used in competitive models.

e) **Shared sampling**: Only individuals with higher shared fitness are combined to favor individuals that are significantly different from the others in a population.

3 Literature Review

The literature review focuses on the main issues addressed in this work. First, Section 3.1 introduces evolutionary-based algorithms for the MDVRP, followed by parallel algorithms for the MDVRP in Section 3.2. Then, Section 3.3 is devoted to known applications of sequential and parallel coevolutionary algorithms.

3.1 Heuristics for the Multi-Depot Vehicle Routing Problem

Evolutionary algorithms (EAs) use a set of candidate solutions, known as a *population*, and heuristic mechanisms to evolve it like *selection* and *reproduction* (also called *genetic operators*). Evolution proceeds from one generation to the next until a stopping condition is satisfied (Engelbrecht, 2007; Luke, 2013). Evolutionary algorithms have been widely used to solve the VRP, as surveyed in Potvin (2009). The main contributions with regard to the MDVRP are reported below.

Genetic algorithms (GAs) are probably the most widely used class of evolutionary algorithms and were applied as well to the MDVRP. A comprehensive survey of different types of GAs for the MDVRP can be found in Karakatič and Podgorelec (2015), while GAs for the MDVRP, among other VRP variants, are also described in Gendreau et al. (2008). With regard to hybrids, a simulated annealing-based solution acceptance criterion is applied after reproduction in Chen and Xu (2008). The Clarke and Wright savings heuristic and the nearest neighbor heuristic are used in Ho et al. (2008) to create initial solutions for the GAs. A combination of simulated annealing, bee colony optimization and GA is also proposed in Liu (2013). Vidal et al. (2012) and Vidal et al. (2014) introduce a powerful hybrid GA using neighborhood-based heuristics and population-diversity management schemes to address many different types of VRPs, including the MDVRP.

Particle Swarm Optimization (PSO) is inspired by the social behavior of agents, such as swarms and birds flock. Individuals are named *particles*, *flying* in the search space according to simple rules that combine local and global information (Engelbrecht, 2007; Luke, 2013). This problem-solving methodology was used in Wenjing and Ye (2010) for solving the MDVRP.

In addition to EAs, some noteworthy heuristics were proposed to solve the MDVRP. Tabu Search (TS) has been used in several contexts. In particular, Renaud et al. (1996) and Cordeau et al. (1997) use this heuristic for some VRP problems including MDVRP. A hybrid *granular* TS algorithm was proposed by Escobar et al. (2014). Those authors introduce the idea of *granular neighborhoods*, in which the search process uses restricted neighborhoods for each customer defined by a *granularity threshold value*.

An adaptive large neighborhood search (ALNS) approach was introduced by Pisinger and Ropke (2007). It was applied to solve five different variants of the VRP, including the MDVRP.

The method developed by Subramanian et al. (2013) combines an exact procedure based on the set partitioning formulation with an Iterated Local Search (ILS). A Mixed Integer Programming solver is used for the exact procedure.

3.2 Parallel algorithms for the Multi-Depot Vehicle Routing Problem

A few parallel algorithms for the MDVRP are reported in the literature. A parallel version of Ant Colony Optimization (ACO) is introduced in Yu et al. (2011). In this work, the MDVRP was simplified through the definition of a single virtual depot, while insuring that the capacity constraint of each vehicle is satisfied. The algorithm was implemented using a distributed coarse-grained environment composed of eight computers each equipped with a Pentium processor (3 GHz with 512 MB RAM).

A parallel iterated Tabu Search (TS) is proposed in Maischberger and Cordeau (2011); Cordeau and Maischberger (2012). In the first paper, some preliminary results are reported on eight classes of VRPs including the MDVRP. The algorithm was run on a cluster made of 128 nodes (each with a 3GHz Xeon processor). The second paper reports results over four VRP variants, including the MDVRP, as well as other variants with time windows.

3.3 Coevolutionary algorithms

In this section, we review different applications of CAs. Note that a discussion about sequential and parallel versions of CAs can be found in Popovici and De Jong (2006). The authors show in particular how different population update strategies can impact the overall performance. The authors empirically demonstrate the superiority of the parallel version over the sequential one on benchmark functions, using a number of different metrics.

A competitive model with three populations is used in Li et al. (2014) to solve constrained design problems. The first population is made of candidate solutions from the design space while the other populations represent disturbances due to uncertainties. Li and Yao (2012) propose a cooperative coevolutionary algorithm in a particle swarm environment. They apply it to large scale optimization problems on functions with up to 2,000 variables. Other methodologies using particle swarm and coevolution can also be found in Aote et al. (2015) and Chen et al. (2010).

A study on sequential and parallel versions of a cooperative coevolutionary algorithms based on the ES(1+1) evolution strategy is presented in Jansen and Wiegand (2003, 2004). The authors identify situations where a cooperative scheme could be inappropriate, like problems involving non separable functions. Depending on the decomposition method or the characteristics of the function, the coevolutionary algorithm could even be harmful.

As far as we know, the coevolutionary paradigm has never been applied to the MDVRP. With regard to vehicle routing in general, a large scale capacitated arc routing problem is addressed in Mei et al. (2014) using a coevolutionary algorithm. In this work, the routes are grouped into different subsets to be optimized and problem instances with more than 300 edges are solved. A multi-objective capacitated arc routing problem is also studied in Shang et al. (2014). A coevolutionary algorithm is presented in Wang and Chen (2013) for a pickup and delivery problem with time windows. To minimize the number of vehicles and the total traveling distance, the authors use two populations: one for diversification purposes and the other for intensification purposes. In the scheduling domain, a competitive coevolutionary quantum genetic algorithm for minimizing the makespan of a job shop scheduling problem is reported in Gu et al. (2010).

4 Cooperative coevolutionary model for the MD-VRP

A cooperative coevolutionary model with problem decomposition is proposed here to solve the MDVRP. The main contribution of this method is the computational efficiency resulting from the decomposition of the problem into subproblems. Each subproblem becomes a single depot VRP and evolves independently in its domain space. The decomposition approach considers the depots separately and assigns a subset of customers to each one. Some overlap is possible, that is, customers might be associated to one or more depots depending on their neighbors (other customers or depots). An evolving population is associated with each subproblem and the (partial) solutions in each population must then cooperate to form a complete solution to the MDVRP. In this section, we explain the decomposition approach. Then, Section 5 describes the evolution process for solving the subproblems.

Each customer is assigned to a depot using the two following rules:

1. The closest depot;

2. The closest depot to the closest neighbor node (if different from the one in rule 1);

When the two rules identify the same depot, the assignment is obvious. When the two rules identify two different depots, then the conflict must be addressed in some way. Figure 3 illustrates how these rules are applied. In the figure, there are three depots (nodes A, B and C) and twenty customers (nodes 1 to 20). For customer 1, rule 1 identifies A as the closest depot. Then, rule 2 identifies the closest neighbor to customer 1 as customer 2, whose closest depot is also A. Thus, customer 1 can only be assigned to depot A. In the figure, all white nodes can only be assigned to their closest depot.

Now, let us consider customer 6. Rule 1 identifies B as the closest depot. Then, rule 2 identifies the closest neighbor to customer 6 as customer 7, whose closest depot is A. Since we do not know if it is better to assign customer 6 to depot A or B, customer 6 is initially assigned to both depots. That is, this

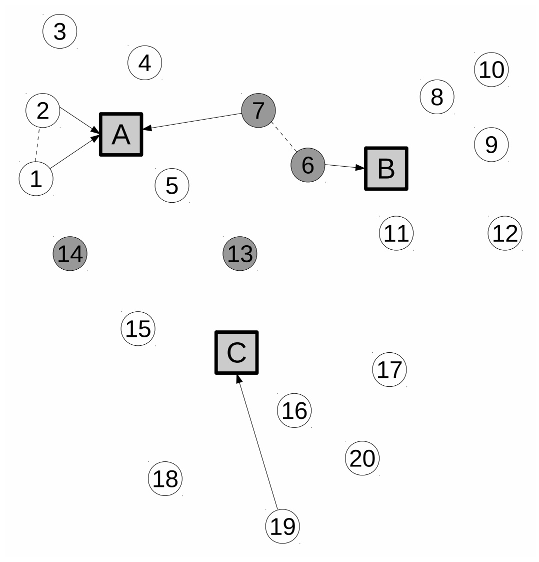


Figure 3: Problem decomposition: Assignment of customers to depots.

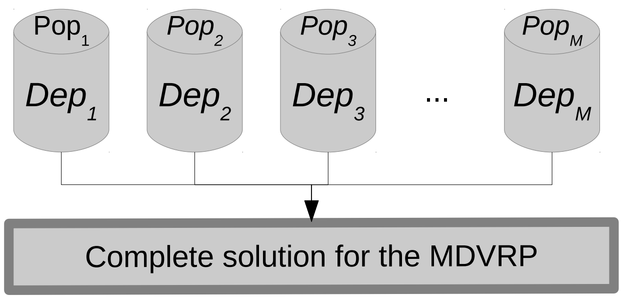


Figure 4: Cooperative coevolutionary model for the MDVRP.

customer will be part of the two evolving populations associated with depots A and B. It implies that there is some overlap among the subproblems. In Figure 3, all gray nodes are assigned to two different depots.

The assignment procedure is further detailed in Algorithm 1. The procedure starts with the set of customers V_C, the set of depots V_D, as well as the set of arcs A. In Line 3 the closest depot m_i to customer i is selected according to rule 1 and inserted in the assignment group A_{m_i} of depot m_i (Line 4). The closest neighbor to customer i is defined as customer j (Line 5) and the closest depot to j, identified as m_j, is selected according to rule 2. The depots are compared in Line 7. If the two depots are different ($m_i \neq m_j$), customer i is also inserted in the allocation group A_{m_j} of depot m_j (Line 8). After processing all customers, the assignment groups are returned (Line 11).

Algorithm 1: Assignment of customers

```
1  assignCustomers($V_C, V_D, A$)
2      for $i \leftarrow 1$ to $N$ do                        // for each customer $i$
           // Rule 1: closest depot
3          $m_i \leftarrow$ getClosestDepot($V_D, i$);
4          insert($A_{m_i}, i$);
           // Rule 2: depot of the closest neighbor
5          $j \leftarrow$ getClosestNeighbor($V_C, i$);
6          $m_j \leftarrow$ getClosestDepot($V_D, j$);
7          if ($m_i \neq m_j$) then                        // Different depots
8              insert($A_{m_j}, i$);
9          end if
10     end for
11     return($A_1 \ldots A_M$);
12 end
```

After this decomposition, each subproblem becomes a classical single depot VRP for a subset of customers identified by the two assignment rules above. Given that the gray nodes are duplicated, a repair operator will be needed to obtain a valid complete solution (see Section 5.5). Each subproblem is solved with an evolutionary algorithm, in which each individual represents a partial solution to the MDVRP. Figure 4 illustrates the structure of the proposed coevolutionary model. For each depot, there is one population which evolves and searches the best routes for the set of customers assigned to it. Then, one individual for each population is selected to create a complete solution for the MDVRP.

A decomposition approach is particularly interesting for problem instances with a low degree of interdependency (coupling) between the subproblems. For example, customer 19 in Figure 3 should clearly be served by depot C. It is unlikely that good solutions will be obtained by assigning this customer to depots A or B, and these solutions are automatically eliminated through the decomposition approach. It is clear that some degree of interdependency exists among the subproblems for the instance illustrated in Figure 3, due to the presence of gray nodes.

This model is strongly suitable for a parallel environment where each population evolves separately and cooperates with other populations to solve the problem. A parallel architecture for this model is proposed in the next section.

5 Parallel evolution strategy

A parallel environment exploiting the Evolution Strategy (ES) paradigm, called *CoES*, supports the evolution of our cooperative coevolutionary model. Evolution Strategy (Engelbrecht, 2007; Freitas et al., 2014; Luke, 2013) is an evolutionary algorithm using *mutation* as the main operator to generate new solutions. ES was chosen because each subproblem has a variable length representation (genotype) and the design of a recombination operator in this case would be rather cumbersome (see Section 5.2).

The proposed parallelization scheme, which is operational under *asynchronous* updates, is shown in Algorithm 2. *CoES* first receives all required information about the problem, in particular the number of customers (N), number of depots (M) and vehicle capacity (Q). Each population is initialized with μ individuals (Line 3), which are encoded using the representation scheme presented in Section 5.1. The initialization procedure uses a semi-greedy method to insert customers from a given list. Parameter α defines the number of customers in this list, as discussed in Section 5.2. Complete solutions are then created with a *Random* fitness sampling strategy (Engelbrecht, 2007) (Line 4). Here, individuals from each initial population are selected randomly to create complete solutions. It should be noted that another strategy is used in the following populations, as explained in Section 5.5. In Line 5, all complete solutions are evaluated and the best one is selected (Line 6). Then, a number of parallel modules are started (Line 7). At the end, the best complete solution to the MDVRP is returned (Line 8).

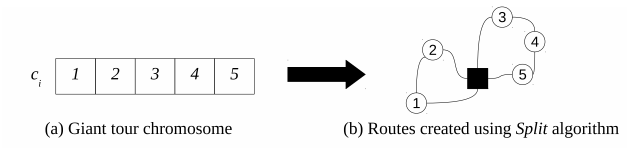
The representation and initialization procedures are described in Sections 5.1 and 5.2. The parallel modules are introduced in Section 5.3.

Algorithm 2: *CoES* – general scheme

```
1 CoES($V_C, V_D, A, N, M, Q$)
2     $A_1 \ldots A_M \leftarrow$ assignCustomers($V_C, V_D, A$);
3     $[P_1 \ldots P_M] \leftarrow$ initializePopulations($A_1 \ldots A_M, \mu, \alpha$) ;
4     $S \leftarrow$ createCompleteSolutions($P_1 \ldots P_M$);
5     $f \leftarrow$ evaluateSolutions($S$);
6     $s* \leftarrow$ getBestSolution();
7     startModules();
8     return($s*$);
9 end
```

5.1 Representation

Individuals from each population are represented by a *giant tour*, without route delimiters. It is basically a single sequence made of all customers assigned to a depot, as shown in Figure 5(a). Since each individual in a population corresponds to a particular depot and subset of customers, the length of the giant tour is likely to change (*variable genotype*). Individual routes are created from this giant tour with the *Split* algorithm (Prins, 2004), which can optimally extract feasible routes from a single sequence. In constrained problems, the *Split* algorithm can be relaxed at the beginning to allow infeasible routes that



c_i
1
2
3
4
5
(a) Giant tour chromosome
(b) Routes created using *Split* algorithm

Figure 5: Giant tour representation and the obtained routes

violate one or more constraints. During the execution of the coevolutionary algorithm, this relaxation is progressively reduced to converge towards feasible routes. Figure 5(b) illustrates two routes that could be obtained from the giant tour representation in 5(a).

5.2 Initialization

The population initialization procedure is shown in Algorithm 3. The first individual in each population is constructed with the Nearest Insertion Heuristic (NIH) (Bodin et al., 1983) while the other ones are constructed with a semi-greedy approach based on NIH. With regard to the first individual, the closest customer to the depot is first inserted in the giant tour. Then, the next customer to be inserted is the one which is closest to the previous one. This is repeated until the giant tour is complete.

Based on this greedy heuristic, a semi-greedy variant generates the remaining individuals. The first customer is selected at random. Then, the remaining customers are sorted based on their distance from the previous one. A restricted candidate list (RCL) is created with the α best-ranked customers and the next customer to be inserted is selected at random in the RCL. This is repeated until the giant tour is complete.

Algorithm 3: Populations initialization

```
1  initializePopulations($A_1 \ldots A_M, \mu, \alpha$)
2      for $i \leftarrow 1$ to $M$ do
3          for $j \leftarrow 1$ to $\mu$ do
4              if ($j == 1$) then // Greedy construction
5                  $ind_j \leftarrow$ greedy($A_i$);
6              else // Semi-Greedy construction
7                  $ind_j \leftarrow$ semiGreedy($A_i$, $\alpha$);
8              end if
9          end for
10         insert($P_i, ind_j$);
11     end for
12 end
```

5.3 Parallel modules and coordination

The parallel modules are executed until a stopping criterion, based on the execution time, is met. Figure 6 depicts the architecture of these modules within *CoES* as well as their communication scheme.

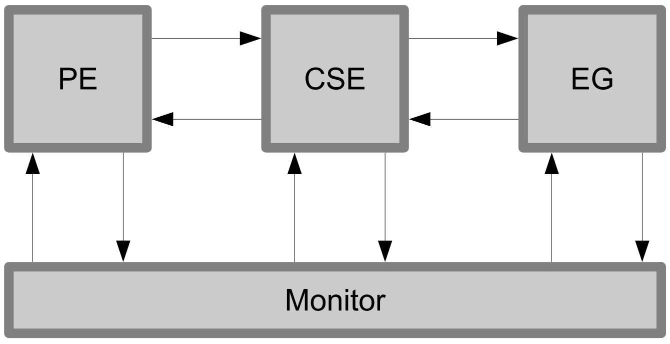


Figure 6: Architecture of the parallel modules in *CoES*.

The *Start Modules* procedure is shown in Algorithm 4. It is called in Line 7 of Algorithm 2 to create a thread for each module and to initialize the environment. A *start* flag is used to indicate that each module should wait until all modules have been initialized. At the beginning of the procedure, the flag is set to *FALSE* (Line 2). At the end, the *start* flag is set to *TRUE* (Line 7) so that all modules can be executed.

Algorithm 4: Start Modules

```
1 startModules()
2     start ← FALSE;
3     createThread(Monitor());
4     createThread(PE());
5     createThread(CSE());
6     createThread(EG());
7     start ← TRUE;
8 end
```

The *Monitor* module manages the parallel processes and transmits information about the MDVRP problem. When the time-based stopping criterion is met, all modules are terminated by the *Monitor* module and the best solution is returned.

The *Population Evolve* (PE) module evolves each population with ES. The *Complete Solutions Evaluate* (CSE) module combines individuals from different populations to create and evaluate complete solutions. In addition, it applies local search heuristics to improve the complete solutions. The *Elite Group* (EG) module maintains an elite set of complete solutions, and also applies local search heuristics to these elite solutions. The various modules mentioned above are explained in detail in the following sections.

5.4 Population Evolve (PE) module

The *Population Evolve* (PE) module manages the ES-based evolution by creating a *thread* for each population. Within a *thread*, the evolution process is run sequentially. This is represented in Figure 7. Note that the ES-based evolution is highlighted in the gray box of Figure 7.

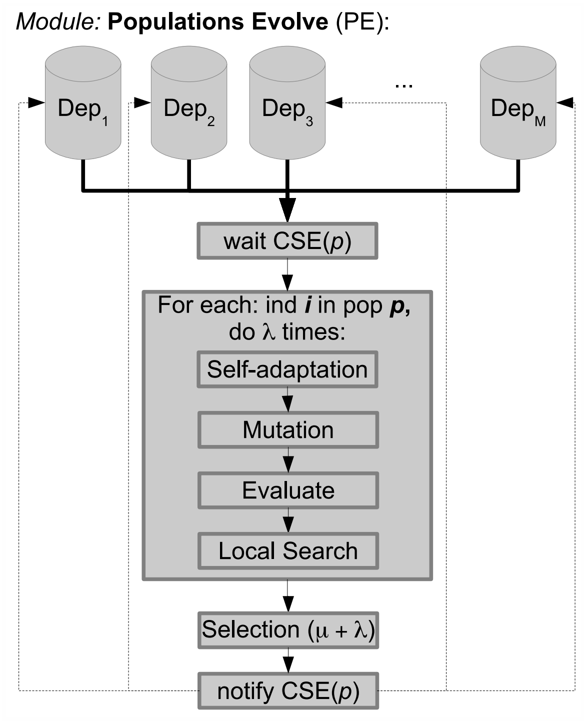


Figure 7: *Population Evolve* module.

With regard to the ES-based population evolution, each one of the λ offspring is generated as follows. First, a parent is selected at random, so that each parent generates λ/μ offspring on average. The self-adaptive procedure updates the number of mutations (strategy parameter σ) using a *binomial distribution* $B(n, p)$. The distribution is computed with n equal to the number of customers in the giant tour (genotype length) and p equal to 0.5. The mutation is applied to the giant tour by selecting two different random positions and by swapping the corresponding customers. This is repeated σ times. Then, the *Split* algorithm is applied and individual routes are created, thus allowing for a local, population-based, fitness evaluation of the newly generated partial solution. With regard to fitness evaluation, the fixed number of vehicles at the depot is accounted for in different ways depending on the status of the parent. Basically, if this number is not exceeded in the parent, then its offspring incur a penalty cost for each vehicle in excess (if any). If this number is already exceeded in the parent, then no penalty is incurred in its offspring. With this approach, a mix of solutions with and without extra vehicles is maintained throughout the search.

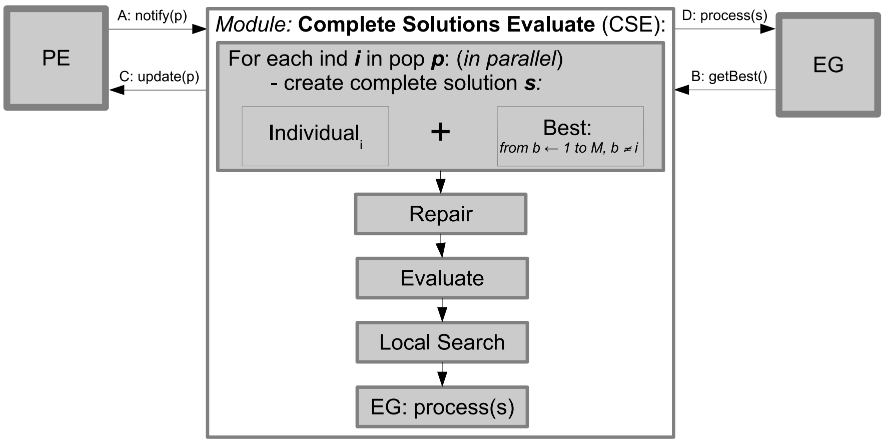

PE
A: notify(p)
C: update(p)
Module: Complete Solutions Evaluate (CSE):
For each ind i in pop p: (in parallel)
- create complete solution s:
Individual i
+
Best: from b ← 1 to M, b ≠ i
Repair
Evaluate
Local Search
EG: process(s)
D: process(s)
B: getBest()
EG

Figure 8: *Complete Solutions Evaluate* module.

A Local Search (LS) procedure is applied to the offspring with probability ρ_{ls}. Nine neighborhood structures are defined to improve the routes. They are the same as those presented in Prins (2004); Siqueira Ruela et al. (2013); Vidal et al. (2012). It is important to note that the local search is performed at the population or subproblem level, therefore only routes starting and ending at the depot and visiting the subset of customers in the subproblem are considered. Let us suppose that u and v are two customer nodes belonging to the same or different routes, x is the successor of u and y is the successor of v along their respective routes. The following moves are applied in random order, and the exploration of the corresponding neighborhood stops as soon as an improving move is found:

- Move 1: move u after v;
- Move 2: move (u, x) after v;
- Move 3: move (x, u) after v;
- Move 4: exchange u and v;
- Move 5: exchange (u, x) with v;

- Move 6: exchange (u, x) and (v, y);
- Move 7: if (u, x) and (v, y) are in the same route (but not adjacent), replace them by (u, v) and (x, y);
- Move 8: if (u, x) and (v, y) are in distinct routes, replace them by (u, v) and (x, y);
- Move 9: if (u, x) and (v, y) are in distinct routes, replace them by (u, y) and (v, x).

Move 7 is a 2-opt move, while moves 8 and 9 are 2-opt* moves. When all neighborhoods have been explored and no improvement to the current solution has been found, the local search procedure stops (Siqueira Ruela et al., 2013).

Once all offspring are created, parents and offspring are ranked according to their local fitness and the ES($\mu + \lambda$) selection is applied to produce the next population. Using a producer-consumer synchronization scheme, the new population is then processed by the CSE module. The corresponding thread waits until the end of the evaluation of all complete solutions before restarting.

5.5 Complete Solutions Evaluate (CSE) module

The *Complete Solutions Evaluate* module receives a population of partial solutions from the PE module and evaluates the global fitness of complete solutions. Each complete solution is created and evaluated in a *multithread* environment, where a *thread* is started for each individual i in population p, as illustrated in Figure 8.

The CSE module is notified by the PE module to process population p at instant A in Figure 8. It combines each individual in p with the best individuals obtained from the other populations to create a complete solution. More precisely, the partial solutions associated with the other depots are taken from the best solution in the *Elite Group* module (*All versus best* strategy), see instant B in Figure 8.

Once a complete solution is created, a repair procedure is applied to remove duplicate customers and to insert customers that are not included in the solution. In both cases, the *least additional cost* heuristic is used (Bodin et al., 1983). The additional cost when customer i is between customers x and y is defined in Equation (12). The least additional cost is looked for when inserting a customer between two consecutive customers in the solution. When duplicates must be removed, only the copy with least additional cost is kept.

$$C_{xiy} = c_{xi} + c_{iy} - c_{xy} \tag{12}$$

To illustrate the removal of a duplicate customer, we refer to Figure 9 where customer i is included in two routes associated with depots A and B. The additional cost is calculated in both cases with Equation (12) to obtain C_{xiy} for depot A and C_{ris} for depot B. If $C_{xiy} < C_{ris}$, then customer i is kept in the route of depot A and the duplicate is removed from the route of depot B.

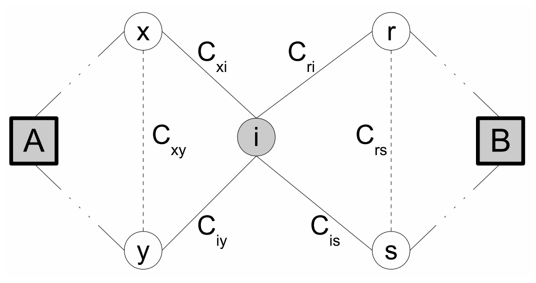


Figure 9: Repair procedure – remove duplicates

When inserting a customer in a solution, all insertion positions between two consecutive customers are considered and the least cost one is chosen. In Figure 10, customer i is inserted between customers x and y in a route of depot A.

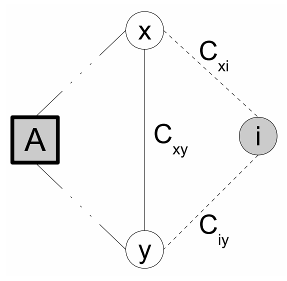


Figure 10: Repair procedure – insert customer

The complete solution is evaluated and submitted to a local search procedure with probability ρ_{ls}. The local search is based on the same moves than those presented in Section 5.4. However, it is now possible for a customer to be moved from one depot to another, thus changing individuals in the populations. This is the reason for the population update at instant C in Figure 8. After the update, the PE module is notified and can restart.

Each complete solution created is sent to the EG module at instant D in Figure 8. The module checks if the complete solution can be part of the elite group, as it is explained next.

5.6 Elite Group (EG) module

The *Elite Group* (EG) module maintains a set of τ elite complete solutions, including the best solution to the MDVRP found so far. This is illustrated in Figure 11. Assuming that a complete solution is submitted by the CSE module, the EG module will try to add the solution to the elite group. If there are less than τ solutions in the elite group, the solution is automatically added.

Otherwise, the new solution can enter the elite group only if it is better than the worst solution in the group, in which case it replaces the latter.

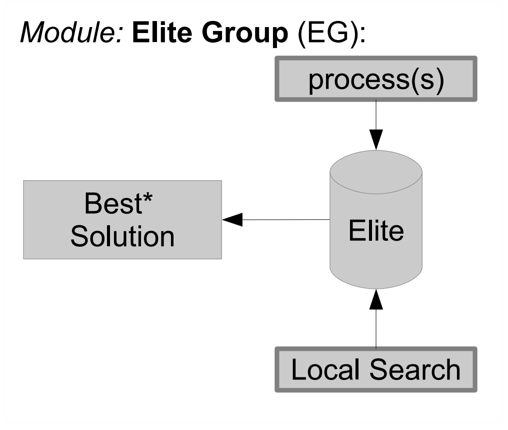


Figure 11: *Elite Group* module.

The local search for the elite group operates in two steps, called PR and RI, where PR stands for Path Relinking and RI for Route Improvement. The local search is run in a *multithread* environment with a *thread* for each solution in the elite group.

PR uses each solution in the elite group as an *initial solution* and the best solution so far as the *guiding solution*. To apply PR in this setting, the difference between an *initial* solution i and the *guiding* solution g must be calculated. We assume that each customer in a solution is characterized by the vehicle serving this customer and the corresponding depot. For each customer, the triplet (customer, depot, vehicle) from the guiding solution is added to set Δ when it differs from the initial solution.

We refer to Figures 12 and 13 for an example. Here, we have an *initial* solution (Figure 12) and a *guiding* solution (Figure 13). There are two depots A and B, twelve customers and two vehicles at each depot (represented by the pentagons numbered 1 and 2). The difference between the initial and guiding solutions corresponds to the nodes in gray, namely customers 2 and 11. Thus $\Delta = \{(2, B, 2), (11, A, 1)\}$. The triplets in Δ are then processed one by one to progressively modify the initial solution. That is, the customer in the first triplet is removed from its route in the initial solution and reinserted in the route indicated by the depot and vehicle of the guiding solution. A new solution is obtained which is closer to the guiding solution. The procedure is then applied again using the new solution and the second triplet. This is repeated until all triplets are done. It is hoped that a solution better than the guiding solution will emerge along the path between the initial and guiding solutions.

The RI improvement procedure is applied to the solution returned by PR. This procedure is the local search mentioned in Section 5.5. If the solution

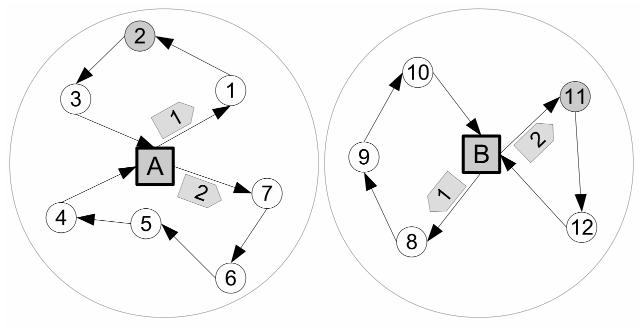


Figure 12: Path Relinking – Initial solution

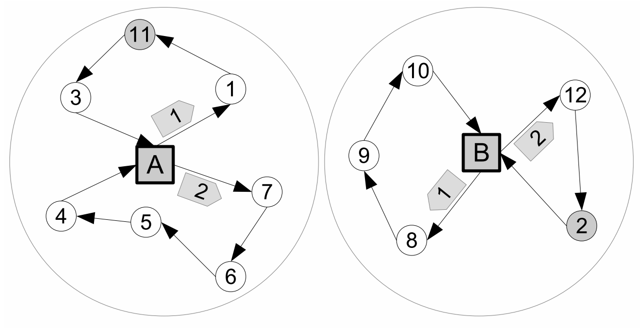


Figure 13: Path Relinking – Guiding solution

obtained at the end is better than the best solution, then the elite group is updated.

6 Computational results

The performance of the proposed CoES algorithm was assessed through a number of computational experiments. They were conducted on the 33 MDVRP benchmark instances taken from the literature (Cordeau et al., 1997; Cordeau and Maischberger, 2012; Vidal et al., 2012; Subramanian et al., 2013; Escobar et al., 2014)[1]. These instances are Euclidean and the time units are the same as the distance units. They are divided into S1 ($P01 - P23$) and S2 ($P24 - P33$)[2] and have various sizes ranging from 48 to 360 customers, see Table 4.

[1]The MDVRP instances might also be obtained in `https://github.com/fboliveira/MDVRP-Instances`

[2]Those instances were proposed by Cordeau et al. (1997) and they are also named *pr01-pr10*.

The algorithm was implemented in C++ 11[3] and all experiments were run on a computer with two 2.50 GHz Intel Xeon (E5-2640) processors with 12 cores per CPU and 96 GB RAM. The computer was run under the Ubuntu 14.04.1 LTS operational system.

A first experiment was realized to set the parameter values, as it is discussed in Section 6.1. A comparison with the best solutions reported in the literature follows in Section 6.2.

6.1 Parameter settings

The algorithm stops when it reaches either the allowed total computation time T_{max} or the maximum computation time without improvement to the best solution T_{upd}. Also, the number of individuals μ in each population and the maximum number of complete solutions in the elite group τ must be defined. Finally, the local search procedure in the PE and CSE modules are applied with a probability ρ_{ls}.

After some preliminary tests and using some values commonly found in the literature, lower and upper bound values were determined for the above parameters. These values are presented in Table 1.

Table 1: Selected parameter values

Parameter	**Lower bound**	**Upper bound**
T_{max}	600 sec.	1,800 sec.
T_{upd}	300 sec.	600 sec.
μ	20	100
τ	5	50
ρ_{ls}	0.2	0.8

As there are two values (*levels*) for each parameter (*factors*), a 2^k factorial experiment was designed to study the impact of each factor as well as their interactions. There are 5 factors with 2 levels each, resulting in 32 observations (2^5) for each instance. Selecting the instance in group S1 ($P01 - P23$) from the benchmark, we obtain a total of 736 observations (i.e., 32×23). Given the large number of factors and total observations, an experiment with a single replication was performed. In this situation, the *Sparsity-of-effects principle* applies, that is, the main effects and low-order interactions usually dominate. Then, we can use an experiment with one replication and combine highest-order interaction for calculating the mean square error (Montgomery and Runger, 2011; Campelo, 2015). The response variable corresponds to the gap between the solution value obtained by CoES and the best-known solution value reported in Vidal et al. (2012).

The experiment is performed with the *Analysis of variance* (ANOVA) method from the *R-Project* (R Core Team, 2015). Among other assumptions, ANOVA

[3] *C++ 2011 Standard*

requires a normal distribution for the residuals. As the number of samples is large, they are approximately normal because of the Central Limit Theorem (CLT) and this assumption is satisfied.

Table 2 reports only the significant factors and interactions identified by ANOVA. In this case, the null hypothesis states that there is no significant impact and a p-value smaller than 0.05 invalidates this assumption. Table 2 shows the significant factors identified by ANOVA along with their corresponding p-value.

Table 2: Significant factors and interactions

Factor/Interaction	***p*-value**
μ	$< 2^{-16}$
τ	0.035
$\mu : \rho_{ls} : \tau$	0.016

Two main factors and one interaction have a significant impact on the results. The most significant factor is the number of individuals in each population μ. The second one is the elite group size τ. The third one corresponds to the interaction of the two previous factors with the local search rate ρ_{ls}. Figure 14 shows box plots of the main factors. Based on these plots, μ was set to 20 and τ was set to 50.

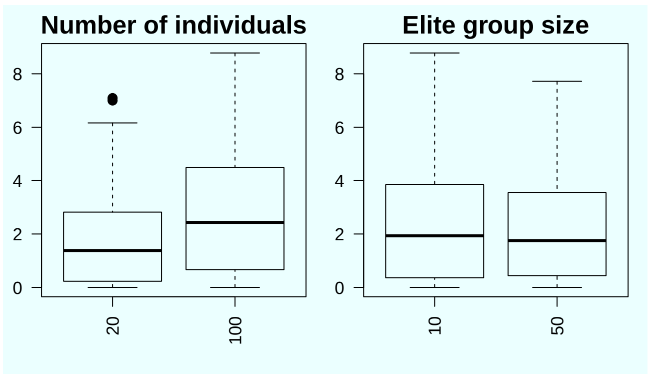


Figure 14: Boxplots of the two main factors

Then, the other parameters can be set by fixing μ and τ to the above values and by observing the interactions. Figure 15 shows boxplots of the interaction of $\mu = 20$ and $\tau = 50$ with ρ_{ls}, T_{max} and T_{upd}, respectively. The parameter values associated with the first two boxplots are shown in the format μ. τ. ρ_{ls}. In the case of the four last boxplots, the format is μ. τ. ρ_{ls}. T_{max} .T_{upd}. The interaction between μ, τ and ρ_{ls} was shown to be a significant factor in Table 2. Figure 15 indicates that an appropriate value for ρ_{ls} is 0.2. With regard to

T_{max} and T_{upd}, Figure 15 shows that suitable values are $T_{max} = 1,800$ and $T_{upd} = 600$.

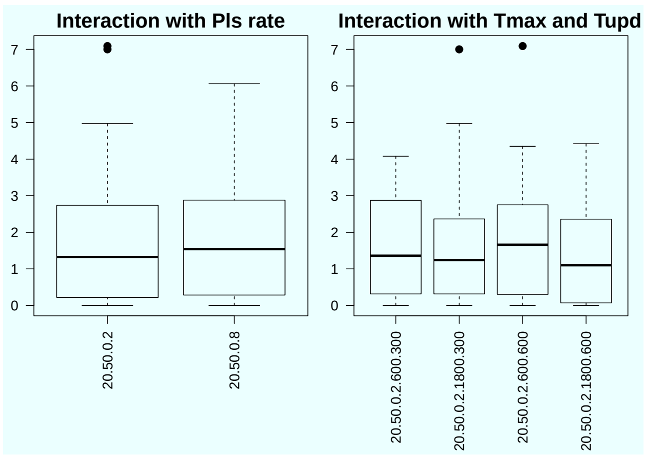


Figure 15: Boxplots of interactions

The final parameter settings for the CoES algorithm are summarized in Table 3. The comparison with other algorithms in the next section is based on these parameter values.

Table 3: Final parameter values

Parameter	**Value**
T_{max}	1,800 sec.
T_{upd}	600 sec.
μ	20
τ	50
ρ_{ls}	0.2

6.2 Comparison with other methods

The performance of our CoES was compared with the best heuristics proposed for the MDVRP, namely, the tabu search (CGL) (Cordeau et al., 1997), the adaptive large neighborhood search (ALNS) (Pisinger and Ropke, 2007), a fuzzy logic guided genetic algorithm (FLGA) (Lau et al., 2010), a parallel iterated tabu search heuristic (ITS) (Cordeau and Maischberger, 2012), a hybrid algorithm combining Iterated Local Search and Set Partitioning (ILS-RVND-SP) (Subramanian et al., 2013), a hybrid genetic algorithm with adaptive diversity control (HGSADC+) (Vidal et al., 2014) and a hybrid Granular Tabu Search (ELTG) (Escobar et al., 2014).

CoES was run 10 times on all benchmark instances, using the previously defined parameter settings. To remove any factor that could impact the performance, the order of execution of all replicates was randomized. Note that heuristics CGL and ELTG were run only once on each instance and the value reported was used to compare with the best value of CoES.

The results from CoES are shown in Table 4. The instance name is in column *Inst*, M is the number of depots, N is the number of customers, Q is the vehicle capacity and T is the maximum duration time of a vehicle route (∞ means that there is no time constraint). The next two columns present the average computation time to reach the best solution (in seconds) and the average solution cost. When (*) appears, the same solution was produced by CoES in each of the 10 runs. The last column shows the best solution cost obtained by CoES.

The best and mean solution values, when compared with the other methods, are shown in Tables 5 and 6, respectively. These tables report the *gap* (in %), that is,

$$100 \times \frac{(Z_{CoES} - Z_{lit})}{Z_{lit}}$$

where Z_{CoES} is the solution value of CoEs and Z_{lit} the value of one of the other methods (as reported in the literature). A negative value indicates that CoES performs better. The *Average* line reports the average *gap* over all instances. The lines *S1* and *S2* show the average *gap* over the instances in each subset. In the Tables, the entries in bold indicate that the same or better solution values were obtained by CoES over the corresponding method.

With regard to the solutions reported in Table 4 labeled with (*), it should be noted that the *gap* with the other methods in Table 6 for those instances is always null ($gap = 0.00$) or negative ($gap < 0.00$). When considering the gap between CoES and the other methods in Table 6 for the mean values, the average over all test instances is always smaller than 1.2%. It is less than 0.9% for subset S1, and less than 1.9% for S2. When considering the gap in Table 5 for the best values, the average over all test instances is always less than 0.6%. It indicates that CoES provides competitive results.

Table 4: CoES($\lambda = \mu$): results

Inst	M	N	Q	T	**Average values**		**Best cost**
					Time (s)	Cost	
P01	4	50	80	∞	1.00	(*) 576.87	576.87
P02	4	50	160	∞	0.50	475.06	473.87
P03	5	75	140	∞	2.50	643.57	641.19
P04	2	100	100	∞	189.70	1011.42	1007.40
P05	2	100	200	∞	26.60	752.39	750.11
P06	3	100	100	∞	77.30	877.86	876.50
P07	4	100	100	∞	24.20	893.36	888.41
P08	2	249	500	310	803.40	4474.23	4450.37
P09	3	249	500	310	513.30	3904.92	3895.70
P10	4	249	500	310	719.90	3680.02	3666.35
P11	5	249	500	310	396.20	3593.37	3569.68
P12	2	80	60	∞	0.90	(*) 1318.95	1318.95
P13	2	80	60	200	0.00	(*) 1318.95	1318.95
P14	2	80	60	180	0.00	(*) 1360.12	1360.12
P15	4	160	60	∞	107.00	2549.65	2526.06
P16	4	160	60	200	8.20	(*) 2572.23	2572.23
P17	4	160	60	180	14.70	2733.80	2709.09
P18	6	240	60	∞	429.10	3781.66	3771.35
P19	6	240	60	200	72.60	(*) 3827.06	3827.06
P20	6	240	60	180	190.20	4094.86	4058.07
P21	9	360	60	∞	554.90	5668.97	5608.26
P22	9	360	60	200	214.00	5708.78	5702.16
P23	9	360	60	180	529.30	6159.90	6129.99
P24	4	48	200	500	0.00	(*) 861.32	861.32
P25	4	96	195	480	370.95	1311.83	1306.44
P26	4	144	190	460	452.48	1812.86	1806.53
P27	4	192	185	440	872.02	2098.21	2060.85
P28	4	240	180	420	399.90	2441.42	2392.31
P29	4	288	175	400	808.95	2780.06	2760.03
P30	6	72	200	500	134.93	1092.70	1089.56
P31	6	144	190	475	343.88	1691.90	1665.01
P32	6	216	180	450	1107.15	2169.92	2150.52
P33	6	288	170	425	216.22	2971.21	2913.66
Average					**290.36**	**2460.89**	**2445.61**
S1					**211.98**	**2694.69**	**2682.55**
S2					**470.65**	**1923.14**	**1900.62**

Table 5: CoES($\lambda = \mu$): comparison with results of the literature – Best values

Instance	CGL	ITS	ILS-RVND-SP	HGSADC+	ELTG
P01	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
P02	**0.00**	0.07	0.07	0.07	0.07
P03	**-0.61**	**0.00**	**0.00**	0.08	**0.00**
P04	0.07	0.64	0.64	0.82	0.64
P05	**-0.43**	0.01	0.01	0.01	0.01
P06	**-0.15**	**0.00**	**0.00**	**0.00**	**0.00**
P07	**-0.40**	0.73	0.73	0.73	0.42
P08	**-0.72**	1.41	1.62	1.77	1.80
P09	**-0.64**	0.90	0.94	0.96	0.38
P10	**-1.30**	0.97	0.96	0.97	1.01
P11	**-0.31**	0.62	0.67	0.67	0.69
P12	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
P13	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
P14	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
P15	**-0.32**	0.82	0.82	0.82	0.82
P16	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
P17	**-0.41**	**0.00**	**0.00**	**0.00**	**0.00**
P18	1.64	1.85	1.85	1.85	1.85
P19	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
P20	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
P21	1.31	2.44	2.44	2.44	2.44
P22	**-0.24**	**0.00**	**0.00**	**0.00**	**0.00**
P23	**-0.16**	0.84	0.84	0.84	0.57
P24	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
P25	**-0.65**	**-0.07**	**-0.07**	**-0.07**	**-0.36**
P26	**-0.50**	0.15	0.15	0.15	0.15
P27	**-1.59**	0.12	0.12	0.07	**-0.16**
P28	**-0.66**	2.62	2.62	2.22	1.82
P29	**-0.29**	3.08	2.96	2.91	1.83
P30	**-0.23**	**0.00**	**0.00**	**0.00**	**0.00**
P31	**-0.67**	0.01	0.01	**0.00**	**-0.03**
P32	**-1.21**	0.81	0.81	0.77	**-0.04**
P33	**-5.70**	1.23	1.37	0.94	0.10
Average	**-0.43**	0.58	0.59	0.58	0.42
S1	**-0.12**	0.49	0.50	0.52	0.47
S2	**-1.15**	0.80	0.80	0.70	0.33

Table 6: CoES($\lambda = \mu$): comparison with results of the literature – Mean values

Instance	**CGL**	**ALNS**	**FLGA**	**ITS**	**ILS-RVND-SP**	**HGSADC+**	**ELTG**
P01	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
P02	0.25	0.32	0.32	0.32	0.32	0.32	0.32
P03	**-0.25**	0.37	0.37	0.37	0.37	0.46	0.37
P04	0.47	0.53	0.98	1.01	1.04	1.08	1.04
P05	**-0.13**	0.01	0.04	0.31	0.29	0.31	0.31
P06	**0.00**	**-0.58**	**-0.55**	0.16	0.16	0.16	0.16
P07	0.16	0.45	0.61	1.29	1.29	1.29	0.98
P08	**-0.18**	1.20	0.81	1.60	1.83	2.07	2.35
P09	**-0.41**	0.32	**-0.29**	0.92	1.05	1.14	0.62
P10	**-0.93**	0.36	0.28	1.17	1.25	1.33	1.39
P11	0.35	0.56	0.32	1.27	1.33	1.30	1.36
P12	**0.00**	**-0.01**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
P13	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
P14	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
P15	0.61	1.19	1.42	1.77	1.77	1.77	1.77
P16	**0.00**	**-0.07**	**-0.24**	**0.00**	**0.00**	**0.00**	**0.00**
P17	0.50	0.91	0.91	0.91	0.87	0.91	0.91
P18	1.92	1.21	1.43	2.13	2.13	2.13	2.13
P19	**0.00**	**-0.30**	**-0.35**	**0.00**	**-0.01**	**0.00**	**0.00**
P20	0.91	0.74	0.78	0.91	0.91	0.91	0.91
P21	2.40	3.04	2.59	3.55	3.55	3.55	3.55
P22	**-0.13**	**-0.23**	**-0.42**	0.12	0.05	0.12	0.12
P23	0.33	1.10	1.00	1.33	1.33	1.31	1.06
P24	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
P25	**-0.24**	0.34	**-0.08**	0.34	0.25	0.34	0.06
P26	**-0.15**	0.38	0.18	0.48	0.49	0.50	0.50
P27	0.19	1.81	1.06	1.81	1.81	1.94	1.65
P28	1.38	4.43	2.09	4.46	4.42	4.73	3.91
P29	0.43	3.44	1.87	3.62	3.53	3.88	2.57
P30	0.05	0.29	0.29	0.29	0.29	0.29	0.29
P31	0.93	1.63	1.52	1.58	1.61	1.63	1.59
P32	**-0.32**	1.57	0.83	1.45	1.62	1.72	0.86
P33	**-3.83**	2.82	1.77	2.93	3.08	3.59	2.08
Average	0.13	0.84	0.59	1.09	1.11	1.18	1.00
S1	0.26	0.48	0.44	0.83	0.85	0.88	0.84
S2	**-0.16**	1.67	0.95	1.70	1.71	1.86	1.35

As HGSADC+ uses an evolutionary approach, some analyses have been performed using the average costs of that method as a reference. Figure 16 shows the average gap of our algorithm with HGSADC+ when the instances are sorted in increasing order of the ratio between the number of customers and the number of depots. The graph presents a subtle trend suggesting that increasing the number of customers in each subproblem could make the problem harder for CoES. However, the correlation factor is $R^2 = 40.20\%$ and needs to be further investigated.

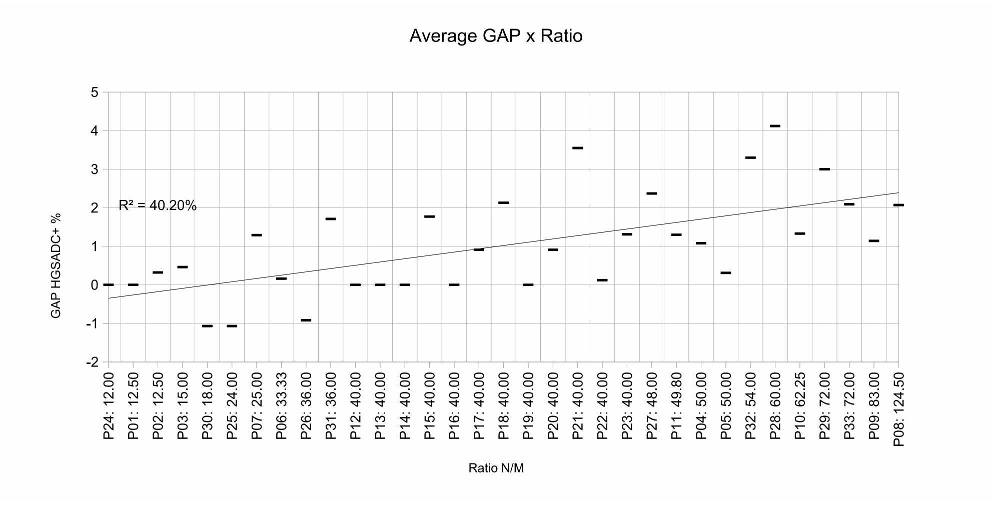


Figure 16: Average gap based on the ratio between customers and depots.

Figure 17 shows the average gap with HGSADC+ when the instances are sorted in increasing order of the ratio between the number of customers assigned to more than one depot and the number of depots. It measures the overlap among the subproblems. The results suggest that the performance of the coevolutionary approach is not directly affected by this characteristic, even for instances in which the overlap is important, e.g., P08 and P33.

Overall, the coevolution paradigm was able to produce competitive solutions, even if its inherent parallel nature was not fully exploited in our implementation.

7 Conclusion and future work

This paper proposed a cooperative coevolutionary algorithm to solve the MDVRP. In this algorithm, each depot is associated with a population with its assigned customers. Individuals in each population represent partial single-depot solutions to the problem. Each population evolves separately, but the quality of an individual depends on its ability to cooperate with partial solutions from other populations to form a good complete solution to the MDVRP.

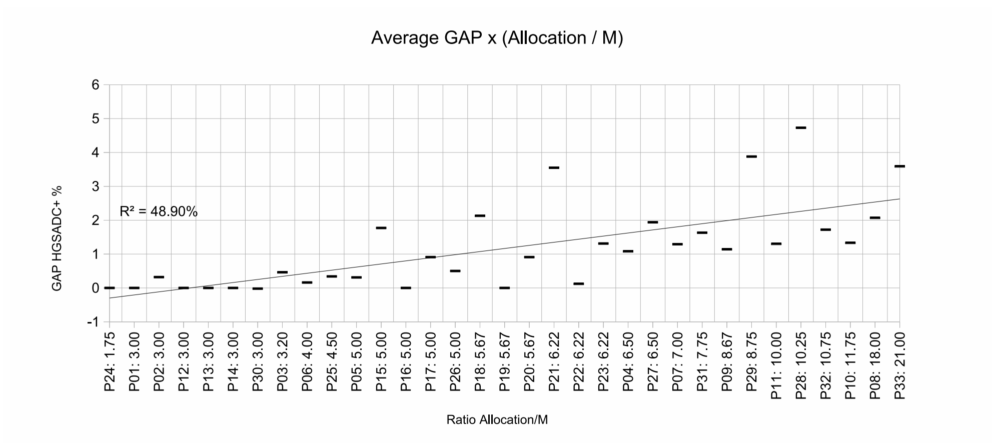


Figure 17: Average gap based on the ratio between customer assignments and depots.

The results show that our coevolutionary algorithm produces competitive solutions when compared with the best known solutions, even improving some of them. The benefit of our approach comes from its ability to decompose complex problems into simpler subproblems and evolve solutions to the subproblems in parallel. The decomposition approach also makes the method more scalable. In large MDVRP instances, it is unlikely that customers close to a depot will be allocated to a distant depot. Therefore, the coevolutionary algorithm incorporates important characteristics of a problem instance and allows a reduction of the search space. Moreover, the evolutionary engine leads to simpler representations and genetic operators. Finally, the coevolutionary algorithm proposed in this work would greatly benefit from cloud computing architectures, cluster computing and GPU programming.

One interesting avenue of research would be to integrate mathematical programming into the local searches (matheuristic). Since each subproblem reduces to a single-depot VRP, the use of mathematical programming within each population could be beneficial. Additionally, other sophisticated local search strategies could be incorporated into the EG module, like Tabu Search. Finally, we intend to improve the parallel implementation by exploiting GPU programming for the PE, CSE and EG modules.

References

Shailendra S. Aote, Mukesh M. Raghuwanshi, and Latesh G. Malik. A new particle swarm optimizer with cooperative coevolution for large scale optimization. In S.C. Satapathy, B.N. Biswal, S. K. Udgata, and J.K. Mandal, editors, *Proceedings of the 3rd International Conference on Frontiers of In-*

telligent Computing: Theory and Applications (FICTA) 2014, volume 327 of *Advances in Intelligent Systems and Computing*, pages 781–789. Springer International Publishing, 2015.

Roberto Baldacci and Aristide Mingozzi. A unified exact method for solving different classes of vehicle routing problems. *Mathematical Programming*, 120 (2):347–380, 2009.

Ivan Blecic, Arnaldo Cecchini, and Giuseppe A. Trunfio. Fast and accurate optimization of a GPU-accelerated CA urban model through cooperative coevolutionary particle swarms. *Procedia Computer Science*, 29:1631 – 1643, 2014.

Lawrence Bodin, Bruce L. Golden, Arjang A. Assad, and Michael O. Ball. Routing and Scheduling of Vehicles and Crews: The State of The Art. *Computers & Operations Research*, 10, 1983.

Felipe Campelo. Lecture notes on design and analysis of experiments. `https://github.com/fcampelo/Design-and-Analysis-of-Experiments`, 2015. Version 2.11, Chapter 12; Creative Commons BY-NC-SA 4.0.

Hanning Chen, Yunlong Zhu, and Kunyuan Hu. Discrete and continuous optimization based on multi-swarm coevolution. *Natural Computing*, 9(3):659–682, 2010.

Haoyang Chen, Yasukuni Mori, and Ikuo Matsuba. Solving the balance problem of massively multiplayer online role-playing games using coevolutionary programming. *Applied Soft Computing*, 18:1 – 11, 2014.

Peiyou Chen and Xinming Xu. A hybrid algorithm for multi-depot vehicle routing problem. In *IEEE International Conference on Service Operations and Logistics and Informatics*, volume 2, pages 2031–2034, 2008.

Jean-François Cordeau and Mirko Maischberger. A parallel iterated tabu search heuristic for vehicle routing problems. *Computers & Operations Research*, 39 (9):2033 – 2050, 2012.

Jean-François Cordeau, Michel Gendreau, and Gilbert Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30 (2):105–119, 1997.

Karl F. Doerner and Verena Schmid. Survey: Matheuristics for rich vehicle routing problems. In MaríaJ. Blesa, Christian Blum, Günther Raidl, Andrea Roli, and Michael Sampels, editors, *Hybrid Metaheuristics*, volume 6373 of *Lecture Notes in Computer Science*, pages 206–221. Springer Berlin Heidelberg, 2010.

Marc Ebner. Coevolution and the red queen effect shape virtual plants. *Genetic Programming and Evolvable Machines*, 7:103–123, 2006.

Andries P. Engelbrecht. *Computational Intelligence: An Introduction.* Wiley, 2007.

John Willmer Escobar, Rodrigo Linfati, Paolo Toth, and Maria G. Baldoquin. A hybrid granular tabu search algorithm for the multi-depot vehicle routing problem. *Journal of Heuristics*, 20(5):483–509, 2014.

Sevan Gregory Ficici. *Solution concepts in coevolutionary algorithms.* PhD thesis, Brandeis University, Waltham, MA, USA, 2004.

Alan Robert Resende Freitas, Frederico Gadelha Guimarães, Rodrigo César Pedrosa Silva, and Marcone Jamilson Freitas Souza. Memetic self-adaptive evolution strategies applied to the maximum diversity problem. *Optimization Letters*, 8(2):705–714, 2014.

Michel Gendreau, Jean-Yves Potvin, Olli Bräysy, Geir Hasle, and Arne Løkketangen. Metaheuristics for the vehicle routing problem and its extensions: A categorized bibliography. In B.L. Golden, S. Raghavan, and E.A. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces*, pages 143–169. Springer US, 2008.

Jinwei Gu, Manzhan Gu, Cuiwen Cao, and Xingsheng Gu. A novel competitive co-evolutionary quantum genetic algorithm for stochastic job shop scheduling problem. *Computers & Operations Research*, 37(5):927 – 937, 2010.

William Ho, George T.S. Ho, Ping Ji, and Henry C.W. Lau. A hybrid genetic algorithm for the multi-depot vehicle routing problem. *Engineering Applications of Artificial Intelligence*, 21(4):548 – 557, 2008.

Thomas Jansen and R. Paul Wiegand. Sequential versus parallel cooperative coevolutionary (1+1) EAs. In *IEEE Congress on Evolutionary Computation*, volume 1, pages 30–37, 2003.

Thomas Jansen and R. Paul Wiegand. The cooperative coevolutionary (1+1) EA. *Evolutionary Computation*, 12(4):405 – 434, 2004.

Sašo Karakatič and Vili Podgorelec. A survey of genetic algorithms for solving multi depot vehicle routing problem. *Applied Soft Computing*, 27:519–532, 2015.

Yoshiaki Katada and Yuta Handa. Tracking the red queen effect by estimating features of competitive co-evolutionary fitness landscapes. In *IEEE Congress on Evolutionary Computation*, pages 1–8, 2010.

Ahmed A. Ladjici and Mohamed Boudour. Nash–Cournot equilibrium of a deregulated electricity market using competitive coevolutionary algorithms. *Electric Power Systems Research*, 81(4):958 – 966, 2011.

Ahmed A. Ladjici, Ahmed Tiguercha, and Mohamed Boudour. Nash equilibrium in a two-settlement electricity market using competitive coevolutionary algorithms. *International Journal of Electrical Power & Energy Systems*, 57: 148 – 155, 2014.

H.C.W. Lau, T. Chan, W.T. Tsui, and W.K. Pang. Application of genetic algorithms to solve the multidepot vehicle routing problem. *IEEE Transactions on Automation Science and Engineering*, 7(2):383–392, 2010.

Min Li, Frederico G. Guimarães, and David A. Lowther. A competitive coevolutionary algorithm for constrained robust design. In *9th IET International Conference on Computation in Electromagnetics*, pages 1–2, 2014.

Xiaodong Li and Xin Yao. Cooperatively coevolving particle swarms for large scale optimization. *IEEE Transactions on Evolutionary Computation*, 16(2): 210–224, 2012.

Chun-Ying Liu. An improved adaptive genetic algorithm for the multi-depot vehicle routing problem with time window. *Journal of Networks*, 8(5), 2013.

Sean Luke. *Essentials of Metaheuristics*. Lulu, second edition, 2013. Available at http://cs.gmu.edu/∼sean/book/metaheuristics/.

Mirko Maischberger and Jean-François Cordeau. Solving variants of the vehicle routing problem with a simple parallel iterated tabu search. In Julia Pahl, Torsten Reiners, and Stefan Voß, editors, *Network Optimization*, volume 6701 of *Lecture Notes in Computer Science*, pages 395–400. Springer Berlin Heidelberg, 2011.

Yi Mei, Xiaodong Li, and Xin Yao. Cooperative coevolution with route distance grouping for large-scale capacitated arc routing problems. *IEEE Transactions on Evolutionary Computation*, 18(3):435–449, 2014.

Douglas C. Montgomery and George C. Runger. *Applied Statistics and Probability for Engineers*. John Wiley & Sons, 5th edition, 2011.

Jairo R. Montoya-Torres, Julián López Franco, Santiago Nieto Isaza, Heriberto Felizzola Jiménez, and Nilson Herazo-Padilla. A literature review on the vehicle routing problem with multiple depots. *Computers & Industrial Engineering*, 79:115 – 129, 2015.

David Pisinger and Stefan Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403 – 2435, 2007.

Elena Popovici and Kenneth De Jong. Sequential versus parallel cooperative coevolutionary algorithms for optimization. In *IEEE Congress on Evolutionary Computation*, pages 1610–1617, 2006.

Jean-Yves Potvin. State-of-the art review—Evolutionary algorithms for vehicle routing. *INFORMS Journal on Computing*, 21(4):518–548, 2009.

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

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2015. URL `http://www.R-project.org/`.

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

Ronghua Shang, Yuying Wang, Jia Wang, Licheng Jiao, Shuo Wang, and Liping Qi. A multi-population cooperative coevolutionary algorithm for multi-objective capacitated arc routing problem. *Information Sciences*, 277:609 – 642, 2014.

André Siqueira Ruela, Frederico Gadelha Guimarães, Ricardo Augusto Rabelo Oliveira, Brayan Neves, Vicente Peixoto Amorim, and Larissa Maiara Fraga. A parallel hybrid genetic algorithm on cloud computing for the vehicle routing problem with time windows. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 2467–2472, 2013.

Anand Subramanian, Eduardo Uchoa, and Luiz Satoru Ochi. A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research*, 40(10):2519 – 2531, 2013.

Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, Nadia Lahrichi, and Walter Rei. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60(3):611–624, 2012.

Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. Implicit depot assignments and rotations in vehicle routing heuristics. *European Journal of Operational Research*, 237(1):15 – 28, 2014.

Hsiao-Fan Wang and Ying-Yen Chen. A coevolutionary algorithm for the flexible delivery and pickup problem with time windows. *International Journal of Production Economics*, 141(1):4 – 13, 2013.

Xingwei Wang, Hui Cheng, and Min Huang. Qos multicast routing protocol oriented to cognitive network using competitive coevolutionary algorithm. *Expert Systems with Applications*, 41(10):4513 – 4528, 2014.

Zhang Wenjing and Jianzhong Ye. An improved particle swarm optimization for the multi-depot vehicle routing problem. In *International Conference on E-Business and E-Government*, pages 3188–3192, 2010.

B. Yu, Z. Z. Yang, and J.-X. Xie. A parallel improved ant colony optimization for multi-depot vehicle routing problem. *Journal of the Operational Research Society*, 62(1):183–188, 2011.