



# A Strength Pareto Evolutionary Algorithm for Solving the Capacitated Vehicle Routing Problem with Time Windows

Wissam Marrouche<sup>1</sup> <sup>a</sup> and Haidar M. Harmanani<sup>2</sup> <sup>b</sup>

<sup>1</sup>*School of Computing, University of Portsmouth, Portsmouth, U.K.*

<sup>2</sup>*Computer Science Department, Lebanese American University, Byblos, Lebanon*

**Keywords:** Capacitated Vehicle Routing Problem with Time Windows, Strength Pareto Evolutionary Algorithm, Hill Climbing, Multi Objective Optimization.


**Abstract:** The capacitated vehicle routing problem with time windows (CVRPTW) belongs to a set of complex combinatorial optimization problems that find major application in logistic systems and telecommunications. It is a complex variant of the well studied capacitated vehicle routing problem (CVRP). Few meta-heuristic approaches have been proposed to solve the CVRPTW problem in literature. In this paper, we examine a population based meta-heuristic algorithm, more precisely the strength Pareto evolutionary algorithm SPEA2 with hill climbing as a local search. The novelty of the approach lies in the choice of multi objective evolutionary algorithm namely SPEA2, its suggested evolutionary operators, and the optimization for three objectives: the number of routes, the total distance travelled, and the average time per route. The proposed algorithm is implemented using Python, and tested on the Solomon's instances benchmark. Favorable results are reported and suggestions for further improvements are discussed.


## 1 INTRODUCTION

The vehicle routing problem (VRP) is a combinatorial optimization problem that arises in major application logistic systems such as during organizing delivery and/or collection of payloads in a given geographical region. It is one of the most studied problem in optimization field and the advancement brought to this problem can benefit the applied operational research and management science (Rizzoli et al., 2004). The plain VRP (Toth and Vigo, 2001) can be stretched out by imposing constraints on one or more variables of the standard problem. The capacitated vehicle routing problem (CVRP) is the classical version, where a limited vehicle capacity is taken into account along with routes optimization. If we relax the multiple route constraint and force all customers to be served by a single route, the CVRP reduces into the standard travelling salesman problem, proving that CVRP is NP-hard (Rizzoli et al., 2004). Furthermore, when the VRP is combined with capacity and time windows constraints, the new variant is known as CVRPTW. Similarly to (Toth and Vigo, 2001), in this work we

consider CVRPTW with multi identical vehicles and single depot while optimizing for certain objectives that we will detail shortly in the next section.

The CVRPTW problem can be tackled using exact methods like integer linear programming (ILP); however this approach is computationally expensive since the problem is known to be an NP-hard (Dash and Anekal, 2014). Furthermore, the time windows establish a series of precedence on visits, which makes the problem asymmetric, even if the distance and time matrices were originally symmetric. In practice, we can rely on heuristics and metaheuristic methods, which provide possible approaches to good solutions for industrial scale problems (Rizzoli et al., 2004). In this context, we tackle the CVRPTW problem using the population-based strength Pareto evolutionary algorithm (SPEA2) introduced by Zitzler et al. (Zitzler, 2001). Here, SPEA2 is hybridized with a hill climbing local search to enhance the convergence to a good solution. The novelty of the work is that it optimizes for three objectives mainly the total distance, the number of routes (equal to the number of vehicles used) and the average time per route which is a new parameter introduced to balance the different vehicles operation time and route lengths.

<sup>a</sup>  <https://orcid.org/0000-0002-0626-1442>

<sup>b</sup>  <https://orcid.org/0000-0001-5416-4383>

The remains of the papers are organised as follows: Section 2 describes mathematically the CVRPTW problem, Section 3 provides a brief listing of VRP related work in the literature and Section 4 offers a comprehensive overview of the methodology for our proposed solution. Section 5 reports and evaluate the simulation results. Finally, Section 6 summarizes the results and makes suggestions for further research.

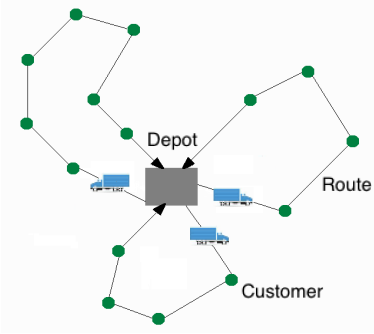


Figure 1: Representation of a typical solution to VRP.

## 2 PROBLEM FORMULATION

Formally, the CVRPTW problem can be stated as follows (Jawarneh and Abdullah, 2015):

Relying on Solomon's 1987 model, the problem formulation presupposes  $N + 1$  customers and  $K$  vehicles. Customer 0 is positioned at the depot node, and each arc in the network constitutes a link between two nodes  $i, j$  and determine the direction of travel with travel time  $t_{ij}$  and cost  $c_{ij}$ . Every route starts from the depot, navigates through a set of customers, and finally goes back to the depot as in Fig 1. Every vehicle is related to one route in the network. In Solomon's 56 CVRPTW 100 customer instances, the speed of the vehicle is considered to be unitary; which means that just one unit of time is needed to traverse one unit of distance. As previously iterated, each vehicle has the same capacity  $q_k$ , and every customer with demand  $m_i$  must be serviced only one time by one of the  $K$  vehicles. The aggregation of the capacity of all the demands on a given route must be less or equal to the  $q_k$  capacity of vehicle  $k$  embarking on this route and no vehicles should be surcharged. The time window restrictions for each node  $i$  are inferred by a predefined time interval, an earliest arrival time ( $e_i$ ), and a latest arrival time ( $l_i$ ). The vehicle must visit the customer before the  $l_i$ ; if it reaches before the  $e_i$ , then it should stand by. Moreover, a service time  $f_i$  for each customer for a given loading and unloading time is taken into consideration. Solomon considers this

loading and unloading time to be unique regardless of the amount of the demands. Vehicles are expected to complete their route within the total route time  $r_k$ . We define for CVRPTW given the subsequent parameters and variables:

- $x_{ijk} = \begin{cases} 0 & \text{if there is no arc from node } i \text{ to node } j \\ 1 & \text{otherwise} \end{cases}$   
 $i \neq j; i, j \in \{0, 1, 2, \dots, N\}; k \in \{1, \dots, K\}$
- $t_i$  arrival time at point  $i$
- $w_i$  waiting time at point  $i$
- $K$  number of vehicles
- $N$  number of customers (0 denotes the central depot)
- $c_{ij}$  cost incurred in arc from customer  $i$  to customer  $j$
- $t_{ij}$  travel time between customer  $i$  and customer  $j$
- $m_i$  demand at customer  $i$
- $q_k$  capacity of vehicle  $k$
- $e_i$  earliest arrival time at customer  $i$
- $l_i$  latest arrival time at customer  $i$
- $f_i$  service time at customer  $i$
- $r_k$  maximum route time for vehicle  $k$

The exact mathematical formulation for a general CVRPTW problem is as follows:

$$\text{Minimise } \sum_{i=0}^N \sum_{j=0, j \neq i}^N \sum_{k=1}^K c_{ij} x_{ijk} \quad (1)$$

Subject the equation to

$$\sum_{k=1}^K \sum_{j=1}^N x_{ijk} \leq K \quad \text{for } i = 0 \quad (2)$$

$$\sum_{j=0}^N x_{ijk} = \sum_{j=1}^N x_{ijk} \leq 1 \quad \text{for } i = 0 \text{ and } k \in \{1, 2, \dots, K\} \quad (3)$$

$$\sum_{k=1}^K \sum_{j=0, j \neq i}^N x_{ijk} = 1 \quad \text{for } i \in \{1, 2, \dots, N\} \quad (4)$$

$$\sum_{k=1}^K \sum_{i=0, i \neq j}^N x_{ijk} = 1 \quad \text{for } j \in \{1, 2, \dots, N\} \quad (5)$$

$$\sum_{i=1}^N m_i \sum_{j=0, j \neq i}^N x_{ijk} \leq q_k \quad \text{for } k \in \{1, 2, \dots, K\} \quad (6)$$

$$\sum_{i=1}^N \sum_{j=0, j \neq i}^N x_{ijk}(t_{ij} + f_i + w_i) \leq r_k \quad \text{for } k \in \{1, 2, \dots, K\} \quad (7)$$

$$t_0 = w_0 = f_0 = 0 \quad (8)$$

$$\sum_{k=1}^K \sum_{i=0, j \neq i}^N x_{ijk}(t_i + t_{ij} + f_i + w_i) \leq t_j \quad \text{for } j \in \{1, 2, \dots, N\} \quad (9)$$

$$e_i \leq (t_i + w_i) \leq l_j \quad \text{for } i \in \{1, 2, \dots, N\} \quad (10)$$

$$c_{ij} = \sqrt{(i_x - j_x)^2 + (i_y - j_y)^2} \quad (11)$$

Equation 1 is the objective function of the CVRPTW problem. Constraint (2) shows the maximum  $K$  routes or vehicles leaving the depot. Equation 3 stresses that every route should start and terminate at the main depot. Equations 4 and 5 guaranty that each customer is serviced only once by exactly one vehicle. Equation 6 describe the capacity and equation 7 specify maximum travel time constraints. Constraints (8), (9), and (10) account for the time window. Equation 11 computes the travel cost, where the Euclidean  $i_x$  is the  $x$  coordinate and  $i_y$  is the  $y$  coordinate of customer  $i$ .

### 3 RELATED WORK

Heuristics such as simulated annealing (SA) and its improvements (Czech and Czarnas, 2002; Yasin and Vincent, 2013), and iterated local search (Yasin and Vincent, 2013) indicate that we can get good CVRPTW solutions within an adequate amount of time. It does a limited exploration of the search space quickly, while meta heuristics abide by the principles of intensification and diversification in their search (Rizzoli et al., 2004). Population based meta heuristic algorithms works well for global searches because they offer global exploration and local exploitation capability over simple heuristics. In this context, population-based and several evolutionary and swarm intelligence-based algorithms like particle swarm optimization (PSO), ant colony optimization (ACO), artificial bee colony algorithm (ABC), for solving VRPTW can outperform simple heuristics (Dixit et al., 2019). In general, bio-inspired meta-heuristics were applied to the basic CVRP problem such as the cuckoo search algorithm which mimics the nesting behavior of the cuckoo bird. The advantage of cuckoo search is that it solves CVRP

in an acceptable time (Xiao et al., 2017). Cuckoo search solves efficiently more elaborate VRP problems such as the VRP with heterogeneous fleet, mixed backhauls (pickup and delivery) and time windows (HVRPMBTW) (Meryem and Abdelmadjid, 2015). Also, the bat algorithm (BA) is a new bio-inspired meta-heuristic based on the echolocation behavior of microbats when probing for their prey in nature. In particular, the BA simulation results show that this approach has a satisfactory performance in solving VRPTW instances (Taha et al., 2017). Not to forget that bio-inspired approaches such as artificial immune systems and neural networks are also considered in literature. For the VRP problem many contributions have been made using hybrid algorithms: using a greedy method for the initial solution followed by tabu search (Udin et al., 2017) or based on ant colony optimization (ACO) algorithm (Rizzoli et al., 2004) with a large neighborhood search (LNS) algorithm (Messaoud et al., 2018). In general, metaheuristics have the tendency to yield good results when hybridized with local search methods (Rizzoli et al., 2004; Dixit et al., 2019), adding specific problem intelligence in the enhancement of the solutions. A hybrid genetic algorithm was proposed in (Vidal et al., 2012) for multi-depot and periodic VRP. A similar published work to our studied problem can be found in (Chiang and Hsu, 2014) and (Hsu and Chiang, 2012), which implements multi objective vehicle routing problem with time windows (MOVRPTW) and optimizes for two concurrent objectives, namely the total distance and the number of vehicles to obtain the set of Pareto optimal solutions. In (Hsu and Chiang, 2012), enhanced reproduction operators are proposed for the CVRPTW problem. An even closer work to this paper is found in (Tan et al., 2006), which implements a hybrid multi objective evolutionary algorithm (MOEA) accommodating the sequence-oriented optimization. Also (Garcia-Najera and Bullinaria, 2011) propose an improved MOEA for the VRPTW optimising mainly for two objectives: the number of routes and the total distance travelled. It also introduces as a third objective the completion or delivery time but does not report in its table result the three objectives explicitly but it rather provides a coverage performance metric analysis, which makes it hard for us to compare it directly to our three objectives results.

The major contribution of this paper is the introduction of strength Pareto evolutionary algorithm SPEA2 to solve for the CVRPTW problem. Also, SPEA2 is improved with a local search namely hill climbing operator to reduce the total distance travelled and therefore ensure a fast convergence. In

fact, (Zitzler, 2001) introduced SPEA2 as an improvement to SPEA. The advantage of the SPEA2 algorithm is that it is an effective evolutionary population search algorithm for obtaining Pareto optimal solutions. SPEA2 is more consistent than NSGA-II in terms of diversity and convergence at the expense of computational complexity (King et al., 2010). In this work, SPEA2 optimizes for three objectives (total distance, number of routes, and average time per route) concurrently to solve the single depot CVRPTW. Reducing the average time per route is an important aspect to consider because it lowers the driver cost time and balances the routes lengths. Another contribution of this paper is that it proposes its own evolutionary operators. Promising results were achieved using this new approach which makes it a good reference for comparison in the presence of 3 objectives.

## 4 SOLUTION METHODOLOGY

Our proposed CVPRTW solution is based on the hybrid SPEA2-local search algorithm as detailed in the following subsections:

### 4.1 Solution Encoding

Based on this extracted information from the Solomon's benchmark 100-customer 56 instances (1987) (see (Dixit et al., 2019) for more details), we populate two data structures namely: Nodes list  $Nodes$ , and the fully connected graph matrix  $G$  of distances between nodes that will help us retrieve the cost  $c_{i,j}$ . Again, each node in  $Nodes$  contains information about position coordinates  $(i_x, i_y)$ , demand  $m_i$ , ready time  $e_i$ , service time  $f_i$  and due date of the delivery  $l_i$ . When scheduling we will enforce that the delivery arrival to a customer  $i$  lies within the time window  $[e_i, l_i]$ . As for the arrival time  $t_i$ , the waiting time at point  $i$   $w_i$  and the travel time between two nodes  $t_{i,j}$  they are computed dynamically during the scheduling process.

A typical solution  $S$  to this problem is represented as an object with the following fields: an array of routes (each  $route_k$  has the list of visited nodes), an average time per route, and a total travelled distance. In fact, the minimization of average completion time per route is linked to work-in-progress inventory management (Li et al., 2015) which can yield balanced routes lengths.

### 4.2 Building the Initial Population using a Greedy Approach

We generate a feasible pool of initial solutions using a greedy algorithm (random seeds, i.e. each time visit the nodes in a different random order). Our proposed greedy algorithm, shown in Fig. 2, ensures that the capacity constraint is met as well as the time windows requirement for the initial solution. A validate function is also implemented to ensure that the solution is feasible in terms of capacity, time windows constraints, and due date. If the number of available vehicle is surpassed, we relax this constraint and consider this greedy solution as valid. The check for the number of vehicle is postponed till the end of the overall simulation where practically in the majority of cases it will be valid.

---

**Greedy algorithm**

```

1:  $nodes \leftarrow$  list of nodes
2:  $S \leftarrow$  some initial empty solution
3:  $VisitFlags[] \leftarrow$  array of flag of visited nodes
4: Shuffle the order of nodes ▷ to have each time different solutions
5: repeat
6:    $Route \leftarrow$  some initial empty route
7:   for all  $n \in nodes$  do
8:     if  $VisitFlags[n] = 0$  then ▷ not visited
9:       Append  $n$  to  $Route$ 
10:    if not (capacity and time window are respected in  $Route$ ) then
11:      Remove  $n$  from  $Route$ 
12:    else
13:       $VisitFlags[n] = 1$ 
14:    Append  $Route$  to  $S$ 
15: until allVisited ( $VisitFlags$ )
```

---

Figure 2: Greedy Algorithm Pseudo Code.

### 4.3 Tweak Operator

The tweak operator yields each time a slightly randomly different candidate solution. It is collectively called a modification procedure and it makes the search exploitative. It was implemented as follows:

Considering a certain solution  $S$ , there are two options for the tweak operator: (a) two randomly selected nodes in  $S$  are within the same route – in that case we just perform a regular swap; (b) two randomly selected nodes  $k$  and  $l$  are in different routes namely route  $i$ , and route  $j$ . This option is more complicated and firstly, we move a node  $k$  of the route  $i$ , to the second route  $j$  and insert it next to the second node  $l$  in the route  $j$ . This tweaked  $S$  or  $S^*$  can decrease the number of routes. More precisely, after applying this tweak, a route  $i$  might become empty so we simply remove it from  $S^*$  as a book keeping measure. Using the validate operator, we check the solution; if it is valid we return  $S^*$  otherwise we roll back to the initial  $S$ , and we repeat this trial a number of times (9 times determined empirically) until feasible. If it fails, we simply return the initial solution  $S$ .



#### 4.4 Recombination Operator

The recombination operator is an “asexual recombination”. One of its advantages is that it explores the search space and avoids the fall into local minima. In fact, the implementation is given as follows:

Given a solution  $S$ , we select randomly two different routes in the solution and perform a fixed-point recombination considering the middle of each route, which results in two new offspring routes. If we want, we then mesh randomly the two offspring routes to obtain a uniform recombination. Using the validate operator we check the new solution  $S^*$  obtained after removing from  $S$  the parent routes and adding the offspring routes; if it is valid we return  $S^*$ , otherwise we roll back to  $S$ , and we repeat the trial a number of times equal to the total number of routes plus a bias until feasible. If it fails, we simply return the initial solution  $S$ . Here is an example in Fig 3.

##### Recombination Operation

- 1: Apply fixed-point recombination after determining the middle of each route  
Parent Route 1: {0-6-7-3-2-8-10-0}  
Parent Route 2: {0-4-1-5-9-0}
- 2: Apply the fixed-point recombination  
Offspring Route1: {0-6-7-3-5-9-0}  
Offspring Route2: {0-4-1-2-8-10-0}
- 3: We then optionally mesh the offspring randomly to obtain multiple point uniform recombination  
Offspring Route1: {0-4-7-2-5-10-0}  
Offspring Route2: {0-6-1-3-8-9-0}

Figure 3: Recombination Example: Fixed Point and Uniform.

#### 4.5 Fuse Operator: Naïve Merge

The advantage of this brute force operator (which is also an asexual recombination) is that if applied with a certain low probability, it will help converge quickly to a lower number of routes specially at early generations of the simulation. Its implementation details can be described as follows:

Given a certain solution  $S$ , we select two random routes and join them together, which results in a new route that we add to the route set of  $S$  after removing the two parents and thus obtain  $S^*$ . Using the validate operator we check this new solution  $S^*$ ; if it is a valid one, we return it otherwise we roll back, and we repeat the trial a number of times equal to the total number of routes plus a bias until feasible. If it fails, we simply return the initial solution  $S$ .

#### 4.6 Hill Climbing Operator

The advantage of this local search operator is that it makes the search converge toward optimal solutions after the application of any of the previous operators. In fact, the steepest ascent hill climbing with replace-

ment (Luke, 2013) is used to reduce the total cost (distance travelled) of a given offspring solution; in other words, to force the evolution of population downhill for this objective. Each time the local search operator is called by hill climbing, it performs either a tweak or a recombination (mutually exclusive) operation with a certain probability to be determined empirically. The rationale behind this can be justified by the fact that we need to change from one type of operator to another to escape local minima. See Fig 4.

##### Steepest Ascent Hill Climbing.

```

1:  $n \leftarrow 25$  ▷ Number of attempts to sample the gradient
2:  $TweakRate \leftarrow 0.8$  ▷ In this case  $RecombinationRate$  is 0.2
3:  $CurrentSol \leftarrow$  Some initial candidate solution
4: repeat
5:    $NewSol \leftarrow Tweak(CurrentSol, TweakRate)$ 
6:   for  $i = 0$  to  $n - 1$  do
7:      $Temp \leftarrow Tweak(CurrentSol, TweakRate)$ 
8:     if  $fitness(Temp) > fitness(NewSol)$  then ▷ fitness is the distance
9:        $NewSol \leftarrow Temp$ 
10:  if  $fitness(NewSol) > fitness(CurrentSol)$  then
11:     $CurrentSol \leftarrow NewSol$ 
12: until we have run out of time
13: return  $CurrentSol$ 

```

Figure 4: Steepest Ascent Hill Climbing Algorithm as a Local Search.

#### 4.7 SPEA2 Fitness Computation

We optimize for the three objectives namely total distance of routes  $D$  (linked to fuel cost), number of routes  $K$  (associated with number of vehicles needed), and average time per route  $T_{avg}$  (related to driver cost and balanced route length). More precisely, we can formulate the objectives as follows: based on equation 1 the total distance  $D$  is given as in equation 12:

$$D = \sum_{i=0}^N \sum_{j=0, j \neq i}^N \sum_{k=1}^K c_{ij} x_{ijk} \quad (12)$$

and based on equation 7 the average time per route is equal to:

$$T_{avg} = \sum_{k=1}^K \sum_{i=1}^N \sum_{j=0, j \neq i}^N x_{ijk} (t_{ij} + f_i + w_i) / K. \quad (13)$$

The problem is reduced to optimizing for  $D, K$ , and  $T_{avg}$ . In other words, the fitness function is a vector-valued function that rely on the Pareto dominance concept and is defined as:

$$f = (D, K, T_{avg}) \quad (14)$$

An objective vector  $f_1$  dominates another objective vector  $f_2$  ( $f_1 \succ f_2$ ) if no component of  $f_1$  is smaller than the corresponding component of  $f_2$  and at least one component is greater (Zitzler et al., 2004).

Each individual  $i$  in the archive  $A_t$  and also in the population  $P_t$  (refer to Fig 6) is given a strength value,

$Strength(i)$ , denoting the number of solutions it dominates as described by equation 15.

$$Strength(i) = \sum_{i,j \in P_t \cup A_t} [i \succ j], \quad (15)$$

where the bracketed notation  $[S]$  is 1 if  $S$  is true and 0 otherwise. Based on this definition, the raw fitness  $f(i)$  of an individual  $i$  is determined as follows (Marrouche et al., 2018):

$$f(i) = \sum_{i,j \in P_t \cup A_t, j \succ i} Strength(j). \quad (16)$$

The raw fitness is evaluated by the strengths of its dominators in the archive and in the population, where the strength of a solution is defined as the number of other solutions in the current population that it dominates. Ties are resolved using a nearest neighbor density estimation function,  $Density(i)$ .

$$Density(i) = \frac{1}{\sigma_i^k + 2}, \quad (17)$$

where  $\sigma_i^k$  is the Euclidean distance of the objective values between a given solution  $i$  and the  $k_{th}$  nearest neighbor of the solution, where  $k = \sqrt{(M_P + M_A)}$ , where  $M_P$  is the population size and  $M_A$  is the archive size. For every individual in the population, we sort the population by distance to that individual, and take the  $k_{th}$  closest individual. The algorithm has a runtime complexity of  $O(M_P^2 \lg M_P)$ , as described in (Marrouche et al., 2018).

## 4.8 SPEA2 Algorithm

SPEA2 is a multiple objective optimization (MOO) algorithm and an evolutionary algorithm in the paradigm of evolutionary computation. The objective of the algorithm is to localize and preserve a front of non-dominated solutions, preferably a set of Pareto optimal solutions. We achieve this goal by constructing an archive as in Fig. 5 and relying on an evolutionary routine as introduced in Section 4.9. We propose the deployment of the following SPEA2 algorithm pseudo code in Fig. 6 based on (Luke, 2013).

## 4.9 Evolve Operator

The SPEA2 requires an evolution mechanism, which we base on four operators, namely tweak, recombine, naïve merge and hill climbing operators applied sequentially given certain probabilities that we will list shortly in Section 4.10 and that we determine empirically. This elaborate and diversified operator exploits and explores the search space mainly without falling into the trap of local minima.

---

**SPEA2 Archive**

```

1:  $P \leftarrow \{P_1, \dots, P_m\}$  ▷ population
2:  $O \leftarrow \{O_1, \dots, O_n\}$  ▷ objective to assess with
3:  $a \leftarrow$  desired archive size
4:  $A \leftarrow$  Pareto non-dominated front of  $P$  ▷ the archive
5:  $Q \leftarrow P - A$ 
6: if  $\|A\| < a$  then
7:   Sort  $Q$  by fitness
8:    $A \leftarrow A \cup$  the  $a - \|A\|$  fittest individuals in  $Q$ , breaking ties arbitrarily
9: while  $\|A\| > a$  do
10:   $Closest \leftarrow A_1$ 
11:   $c \leftarrow$  index of  $A_1$  in  $P$ 
12:  for all individual  $A_i \in A$  except  $A_1$  do
13:     $l \leftarrow$  index of  $A_i$  in  $P$ 
14:    for  $k = 1$  to  $m-1$  do
15:      if  $DistKthNearest(k, P_l) < DistKthNearest(k, P_c)$  then
16:         $Closest \leftarrow A_i$ 
17:         $c \leftarrow l$ 
18:      break
19:    else
20:      if  $DistKthNearest(k, P_l) > DistKthNearest(k, P_c)$  then
21:        break
22:   $A \leftarrow \{Closest\}$ 
23: return  $A$ 

```

---

Figure 5: SPEA2 Archive Construction Algorithm (Luke, 2013).

---

**SPEA2 Algorithm**

```

1:  $m \leftarrow$  desired population size
2:  $a \leftarrow$  desired archive size ▷ typically  $a = m$ 
3:  $P \leftarrow \{P_1, \dots, P_m\}$  ▷ build initial population using a greedy method Fig. 2
4: repeat
5:   AssessFitness( $P$ )
6:    $P \leftarrow P \cup A$ 
7:    $BestFront \leftarrow$  Pareto Front of  $P$ 
8:    $A \leftarrow$  Construct SPEA2 Archive of size  $a$  from  $P$ 
9:    $P \leftarrow$  Evolve( $A$ ), using tournament selection of size 3
10: until we have run out of time
11: return  $BestFront$ 

```

---

Figure 6: SPEA2 Algorithm Pseudo Code based on (Luke, 2013).

Table 1: Parameters settings of SPEA2 for C1, RC1 benchmarks.

Parameter	# Gen	Tweak	Recombine	Fuse	popsiz	HC	Bias
Values+	260	0.8	0.4	0.1	200	25	0

## 4.10 Setting of Parameters

For the benchmarks we tried fixed-point and uniform recombination and we selected the better of the two reported solutions. We repeated the experiment tentatively 18 times for each recombination type (fixed point and uniform). Two main groups of configurations values were identified. In fact, TABLE 1 is more exploratory than TABLE 2, which tends to be more exploitative, since a bias of zero implies less number of iterations for the deterministic operators. Exceptionally, for the R1 benchmark some results were better obtained using the first configuration while others the second, so we ended up choosing the better solution among the two. As for hill climbing the distribution of tweak operator ratios are: 0.8 tweak and 0.2 recombination (to escape local minima). All the configuration parameters were determined empirically relying on a step wise approach, i.e. changing one parameter at a time while fixing others.

Table 2: Parameters settings of SPEA2 for C2, RC2 benchmarks.

Parameter	# Gen	Tweak	Recombine	Fuse	popsiz	HC	Bias
Values*	150	0.8	0.4	0.1	200	25	10

Table 3: Results of Hybrid SPEA2 Applied to Solomon's 100-customer benchmark.

Benchmark	Ours			BCO	
	Distance	Routes	Avg Time	Distance	Routes
C101	<b>828.94</b>	10	982.89	828.94	10
C102	864.04	10	987.26	<b>828.94</b>	10
C103	<b>828.94</b>	10	982.89	835.71	10
C104	<b>824.78</b>	10	995.40	885.06	10
C105	866.23	11 $\nabla$	916.78	<b>828.94</b>	10
C106	833.24	10	983.32	<b>828.94</b>	10
C107	836.76	10	983.67	<b>828.94</b>	10
C108	831.98	10	983.20	<b>831.73</b>	10
C109	<b>828.94</b>	10	982.89	840.66	10
C201	<b>591.56</b>	3	3197.19	591.56	3
C202	<b>591.56</b>	3	3197.19	591.56	3
C203	659.72	4 $\nabla$	2992.22	<b>593.21</b>	3
C204	667.84	4 $\nabla$	2561.31	<b>606.90</b>	3
C205	<b>588.88</b>	3	3196.29	588.88	3
C206	<b>588.49</b>	3	3196.16	588.88	3
C207	<b>588.29</b>	3	3220.13	590.59	3
C208	<b>588.49</b>	3	3196.16	593.15	3
R101+	1669.81	20 $\nabla$	180.85	<b>1643.18</b>	20
R102*	1499	18	184.37	<b>1476.11</b>	18
R103+	<b>1240.08</b>	15 $\nabla$	196.46	1245.86	14
R104+	<b>1026.47</b>	12 $\nabla$	201.04	1026.91	11
R105+	1391.59	16 $\nabla$	179.46	<b>1361.39</b>	15
R106+	<b>1261.06</b>	14 $\nabla$	179.95	1264.50	13
R107+	1127.84	12	195.12	<b>1108.11</b>	11
R108+	<b>990.62</b>	10	205.77	994.68	10
R109+	1197.33	12	193.51	<b>1168.91</b>	13
R110+	1125.04	13 $\nabla$	182.92	<b>1108.22</b>	12
R111+	1090.52	12	195.66	<b>1080.84</b>	12
R112*	<b>982.91</b>	11	193.94	992.22	10
R201	1260.58	6 $\nabla$	766.38	<b>1197.09</b>	7
R202	1097.58	6 $\nabla$	837.05	<b>1092.22</b>	6
R203	<b>962.40</b>	6 $\nabla$	730.76	983.06	5
R204	<b>807.29</b>	4 $\nabla$	695.79	845.30	4
R205	1036.02	5	719.31 $\nabla$	<b>999.54</b>	5
R206	<b>941.09</b>	4 $\nabla$	752.71	955.94	4
R207	<b>892.61</b>	4 $\nabla$	715.75	903.59	4
R208	805.50	3	688.20	<b>769.96</b>	3
R209	938.48	5 $\nabla$	640.27	<b>935.57</b>	5
R210	<b>981.40</b>	5 $\nabla$	793.38	988.34	6
R211	<b>854.82</b>	5 $\nabla$	630.22	867.95	3
RC101	1682.02	17 $\nabla$	192.15	<b>1637.40</b>	15
RC102	1532.44	15 $\nabla$	196.52	<b>1486.85</b>	14
RC103	1356.21	12	210.34	<b>1299.38</b>	12
RC104	<b>1190.46</b>	11	210.5	1200.60	10
RC105	1591.59	16 $\nabla$	198.98	<b>1535.80</b>	15
RC106	1441.08	14 $\nabla$	198.36	<b>1403.07</b>	13
RC107	1291.70	13 $\nabla$	197.03	<b>1230.32</b>	12
RC108	<b>1156.92</b>	11	212.29	1165.17	11
RC201	1345.36	6 $\nabla$	748.61	<b>1315.57</b>	7
RC202	1174.23	6 $\nabla$	716.13	<b>1169.72</b>	5
RC203	1038.97	5 $\nabla$	777.83	<b>1010.74</b>	4
RC204	<b>883.45</b>	4 $\nabla$	712.01	890.28	4
RC205	1237.81	7 $\nabla$	721.89	<b>1221.28</b>	7
RC206	1147.15	6 $\nabla$	702.32	<b>1097.65</b>	6
RC207	1061.85	5 $\nabla$	672.37	<b>1024.17</b>	5
RC208	865.07	5 $\nabla$	612.30	<b>864.56</b>	4

\* We list our best solution among the recombination methods (fixed and uniform)

$\nabla$  a solution exists where the number of routes is lower than the reported given a larger total distance

## 5 RESULTS

We implemented a hybrid SPEA2 approach that yielded favorable simulation results reported in TABLE 3 using Python scripting language where each instance is ran on a computing node of the Dell C6220 server. Each node has an Intel Xeon CPU E5-2650, 64 bit architecture dual socket, 8 core's each. We report in the aforementioned table for each benchmark the solution with the lowest total distance belonging to one of the 36 Pareto fronts (since 36 runs), knowing that in many cases there exists a lower number of routes corresponding to a higher total distance value than the reported one. Time wise, running an instance took roughly one day. Our main goal is to optimize

for three objectives rather than the total distance and number of routes as a main target. To our best knowledge, we are not aware of any study of instances with these three objectives (specifically the average time per route objective), therefore the results cannot be exactly compared.

Nonetheless, to have a slightly meaningful reference to compare our obtained results with, we matched our results with the two objective based CVRPTW problems. In fact, if we compare our solution with the absolute best reported results for two objectives in literature (Bychkov and Batsyn, 2018) and (Hsu and Chiang, 2012) we will obtain either optimal or sub optimal solutions. But for more practical matter we compared our results to the bee colony optimisation (BCO) algorithm introduced by (Jawarneh and Abdullah, 2015) where we identified a noticeable improvement in 43 % of the cases as in TABLE 3.

The overall performance of our solution is good since with few configurations we were able to get good results for all benchmarks. In general, the configuration of parameters in TABLE 2 (which is more exploitative) gave better solutions than the TABLE 1 (which is more exploratory) if the best reported number of routes is relatively small (below 9 routes). On the other hand, the first configuration gave mainly better results than the second configuration if the best reported number of routes is large. Some exceptions were observed where configuration 2 gave better results for given benchmarks of type R1.

While executing all the simulations, we clearly noticed that the second configuration is faster in converging to a desired solution than the first one.

Benchmarks C1, R1 and RC1 have a relatively short scheduling horizon and permit only a few customers per route (around 5 to 10) as in Fig 7. On the contrary, the benchmarks R2, C2 and RC2 have a long scheduling horizon permitting many customers (sometimes more than 30) to be visited by the same vehicle as in Fig 8. This is in complete match with the literature (Minocha et al., 2011).

Fig. 9, Fig. 10, Fig. 11 and Fig. 12 show the variance analysis of two benchmarks namely Solomon\_100 R108 and RC201 with fixed and uniform recombination operator. We picked two benchmarks belonging to two different families of benchmarks. The box plot depicts the total distances over 18 instances. The experiment is repeated 36 times (2x18 times) for each benchmark. The overall distribution of the total distance data and its skewness over the instances are fairly close, which indicate that the experiment is repeatable. Also, we can notice that the R108 benchmark is slightly less repeatable than the R201 benchmark. Moreover, no major difference

**RC102 Sample Solution**

Total Distance : 1532.44  
 Average Time Per Route : 196.52  
 Route 1 (Time= 231.32) : 0-92-50-27-29-31-34-93-94-80-0  
 Route 2 (Time= 219.50) : 0-83-19-23-18-48-21-25-24-0  
 Route 3 (Time= 237.41) : 0-39-36-40-38-41-43-35-37-72-0  
 Route 4 (Time= 226.72) : 0-98-73-79-7-46-4-2-100-70-0  
 Route 5 (Time= 235.49) : 0-33-26-28-30-32-89-0  
 Route 6 (Time= 228.24) : 0-65-52-99-57-86-74-77-0  
 Route 7 (Time= 167.06) : 0-64-22-49-20-0  
 Route 8 (Time= 100.24) : 0-90-0  
 Route 9 (Time= 133.83) : 0-42-44-61-81-0  
 Route 10 (Time= 175.13) : 0-85-63-76-51-84-56-66-0  
 Route 11 (Time= 218.26) : 0-69-88-53-78-17-13-0  
 Route 12 (Time= 180.00) : 0-1-3-45-5-8-6-55-68-0  
 Route 13 (Time= 191.93) : 0-12-14-47-16-15-9-10-60-0  
 Route 14 (Time= 222.19) : 0-82-11-87-59-97-75-58-0  
 Route 15 (Time= 180.52) : 0-91-95-62-67-71-54-96-0

Figure 7: RC102 Typical Solution.

**RC202 Sample Solution**

Total Distance : 1174.23  
 Average Time Per Route : 716.13  
 Route 1 (Time= 894.16) : 0-91-95-85-63-33-34-31-29-27-26-28-30-32-89-24-25-77-58-52-0  
 Route 2 (Time= 929.12) : 0-69-98-88-53-99-57-86-87-9-10-59-97-75-74-13-17-60-100-70-96-93-94-80-0  
 Route 3 (Time= 775.57) : 0-82-14-47-16-15-11-12-78-73-79-7-6-8-46-4-43-35-37-0  
 Route 4 (Time= 610.80) : 0-65-83-64-19-23-21-48-18-76-51-84-49-22-20-56-66-0  
 Route 5 (Time= 670.14) : 0-2-45-5-3-1-42-39-36-44-40-38-41-72-54-68-55-0  
 Route 6 (Time= 417.00) : 0-92-62-50-67-71-81-61-90-0

Figure 8: RC202 Typical Solution.

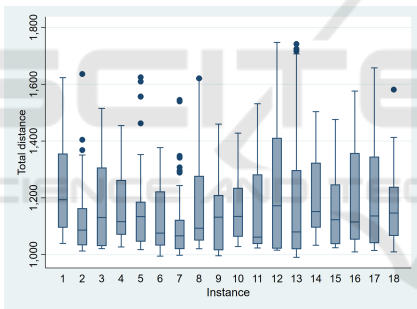


Figure 9: Variance analysis for Solomon\_100 R108 with a uniform recombination operator.

in the distribution and skewness is noticed between fixed and uniform recombination options.

For illustration purposes, we select the Pareto front (one out of the 36 instances) containing the lowest total distance point for the benchmark C201 (size 100). A typical Pareto front surface (with 3 objectives) is shown in Fig. 13, where we apply a triangulation to the Pareto points and plot it as a triangular surface. Here in this example, the number of routes varies between 3 and 19, the total distance between 591.56 and 3840, and the average time between 1759.43 and 3197.19. The standard deviation for total distance is 888.18, for the number of routes it is 4.06 and for the average time per route it is 281.64. In general, we observe a positive correlation between the number of routes and the total distance and an inverse correlation between the number of routes and average

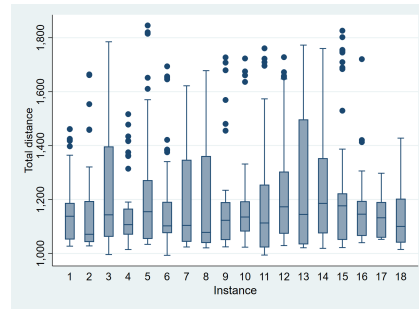


Figure 10: Variance analysis for Solomon\_100 R108 with a fixed recombination operator.

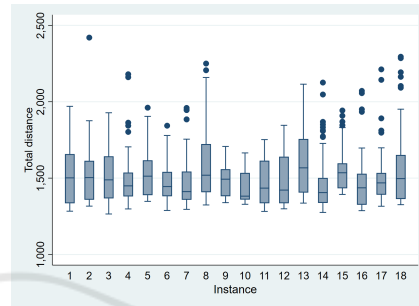


Figure 11: Variance analysis for Solomon\_100 R201 with a uniform recombination operator.

time. The hypervolume (HV) indicator is a measure for comparing Pareto sets in term of the closeness and the spread of the solutions with respect to the Pareto front. For Fig. 13, a good value of  $HV = 0.85$  is obtained given a reference point  $r=1.3$  (30% larger than the nadir point) (Ishibuchi et al., 2018).

Lastly, the visualization of the obtained solutions is very helpful in presenting the final results. Fig. 14, Fig. 15, and Fig. 16 are sample solution depictions of benchmark or more precisely the topological maps (with size 100). More explicitly, C104 has a total distance of 824.78, a number of routes equals to 10, and an average time of 995.40. In the comparison of

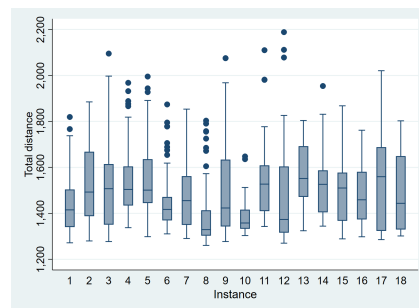


Figure 12: Variance analysis for Solomon\_100 R201 with a fixed recombination operator.



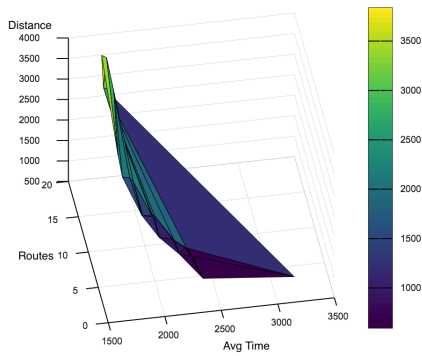


Figure 13: A visual representation sample of the Pareto front surface obtained for Solomon\_100 C201.

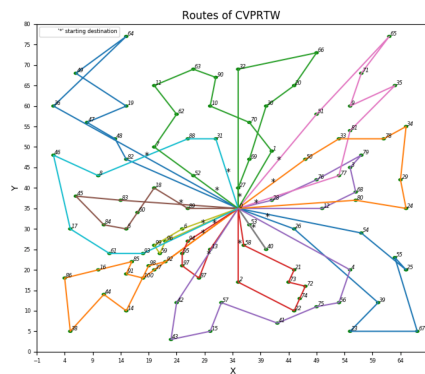


Figure 16: Network topology for Solomon\_100 R103: distance = 1240.08, routes = 15, time = 196.46.

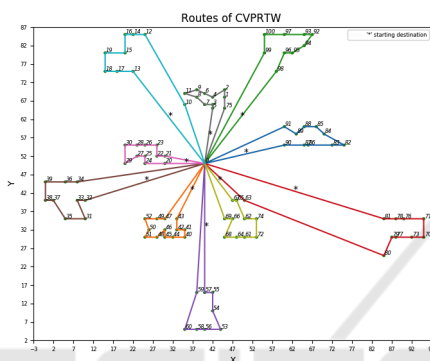


Figure 14: Network topology for Solomon\_100 C104: distance = 824.78, routes = 10, time = 995.40.

these 3 benchmarks, C104 is situated in the middle in terms of total distance, number of routes and average time. C201 has the shortest total distance of 591.56, a number of routes equals to 3, and average time of 3197.19. Clearly, in C201 we can see the reduced number of routes and the short distance they cover since the nodes on the routes are juxtaposed. Whereas R103 has the highest total distance of 1240.08 which is justifiable since the number of routes is elevated and equals to 15 and it covers higher distance between

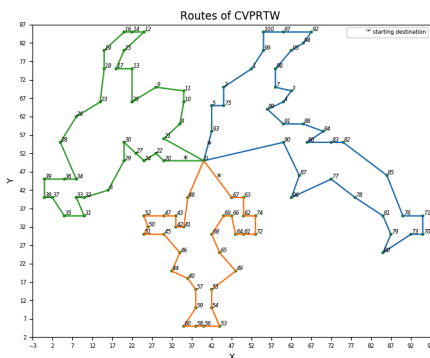


Figure 15: Network topology for Solomon\_100 C201: distance = 591.56, routes = 3, time = 3197.19.

nodes on the routes. As for its average time it is equal to 196.46. Concerning the average times, not much can be inferred from the plots since the problem is asymmetric but one can clearly reiterate the inverse correlation between the number of routes and the average time.

## 6 CONCLUDING REMARKS

In this work, we propose a novel approach using a hybrid SPEA2 augmented with a hill climbing operator to solve the CVRPTW problem optimizing for three objectives: the total distance travelled, the number of routes and the average time per route. The last objective is deemed necessary to reduce driver cost time and balance the routes. Satisfactory results are reported when running the widely studied Solomon's 56 100-customer instances benchmark (a 43% improvement when compared to a main stream bi-objective BCO algorithm). In fact, considering three objectives improves globally the best two standard objectives since it guides the meta heuristic search toward favorable search areas. We checked our proposed algorithm for the bi-objectives case and the results do not rebut this claim. On the other hand, the local search is necessary to ensure quality results despite the latency introduced. The complexity added to the solution in order to improve quality of the solution is in accordance with the paradox: quality results versus speed of convergence (Reimann et al., 2003). As for the simplicity measure, the algorithm is not very hard to configure across different type of classes for this benchmark. Concerning the flexibility of the proposed algorithm, future work should examine scaling this algorithmic solution to handle very large benchmarks or very deploying it on different variants of VRP such as: the green version of the CVRPTW that

accounts for carbon emissions (Lin et al., 2014), or VRP with soft timing constraints, which if violated we must account for a penalty. Ambitious future work can also account for traffic, i.e. higher cost routes in city centers (Rizzoli et al., 2004) in what is known as time dependent VRPTW. Another extension could be to target split delivery vehicle routing problem with time windows (SDVRPTW) where the delivery to a customer can be split between any number of vehicles resulting in considerable improvements in terms number of vehicles used and cost (Da Silva et al., 2014; Feillet et al., 2006).

On the other hand, this work can be generalized to address a different class of VRP altogether, namely the VRP with backhauls (VRPB) (Toth and Vigo, 2001). Lastly, it is also interesting for future work to add problem knowledge to the reproduction operators such as the one proposed in (Hsu and Chiang, 2012).

## ACKNOWLEDGEMENTS

We thank dearly Dr. Janka Chlebkova and Dr. Mohamad Bader-El-Den from the School of Computing at University of Portsmouth for their valuable support. Numerical computations were done on the Sciama High Performance Compute (HPC) cluster which is supported by the ICG, SEPNet and the University of Portsmouth.

## REFERENCES

- Bychkov, I. and Batsyn, M. (2018). A hybrid approach for the capacitated vehicle routing problem with time windows. In *OPTA-SCL*. CEUR Workshop Proceedings.
- Chiang, T.-C. and Hsu, W.-H. (2014). A knowledge-based evolutionary algorithm for the multiobjective vehicle routing problem with time windows. *Computers & Operations Research*, 45:25–37.
- Czech, Z. J. and Czarnas, P. (2002). Parallel simulated annealing for the vehicle routing problem with time windows. In *Proceedings 10th Euromicro workshop on parallel, distributed and network-based processing*, pages 376–383. IEEE.
- Da Silva, M. D. M., Subramanian, A., and Ochi, L. S. (2014). An iterated local search heuristic for the split delivery vehicle routing problem. In *ROADEF-15ème congrès annuel de la Société française de recherche opérationnelle et d'aide à la décision*.
- Dash, M. and Anekal, B. I. (2014). Variables reduction in vehicle routing problems. *SPC ERA IJBM*, 2(6).
- Dixit, A., Mishra, A., and Shukla, A. (2019). Vehicle routing problem with time windows using meta-heuristic algorithms: a survey. In *Harmony search and nature inspired optimization algorithms*, pages 539–546. Springer.
- Feillet, D., Dejax, P., Gendreau, M., and Gueguen, C. (2006). Vehicle routing with time windows and split deliveries. *Technical Paper*, 851.
- Garcia-Najera, A. and Bullinaria, J. A. (2011). An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows. *Computers & Operations Research*, 38(1):287–300.
- Hsu, W.-H. and Chiang, T.-C. (2012). A multiobjective evolutionary algorithm with enhanced reproduction operators for the vehicle routing problem with time windows. In *2012 IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE.
- Ishibuchi, H., Imada, R., Setoguchi, Y., and Nojima, Y. (2018). How to specify a reference point in hypervolume calculation for fair performance comparison. *Evolutionary computation*, 26(3):411–440.
- Jawarneh, S. and Abdullah, S. (2015). Sequential insertion heuristic with adaptive bee colony optimisation algorithm for vehicle routing problem with time windows. *PLoS one*, 10(7):e0130224.
- King, R. A., Deb, K., and Rughooputh, H. (2010). Comparison of nsga-ii and spea2 on the multiobjective environmental/economic dispatch problem. *University of Mauritius Research Journal*, 16(1):485–511.
- Li, W., Dai, H., and Zhang, D. (2015). The relationship between maximum completion time and total completion time in flowshop production. *Procedia Manufacturing*, 1:146–156.
- Lin, C., Choy, K. L., Ho, G. T., Chung, S. H., and Lam, H. (2014). Survey of green vehicle routing problem: past and future trends. *Expert systems with applications*, 41(4):1118–1138.
- Luke, S. (2013). Essentials of metaheuristics. lulu, 2013. *Metaheuristics in large-scale global continuous optimization: A survey*, available at: <http://cs.gmu.edu/~sean/book/metaheuristics>.
- Marrouche, W., Farah, R., and Harmanani, H. M. (2018). A strength pareto evolutionary algorithm for optimizing system-on-chip test schedules. *International Journal of Computational Intelligence and Applications*, 17(02):1850010.
- Meryem, B. and Abdelmadjid, B. (2015). Resolving a vehicle routing problem with heterogeneous fleet, mixed backhauls and time windows using cuckoo behaviour approach. *International Journal of Operational Research*, 24(2):132–144.
- Messaoud, E., El Idrissi, A. E. B., and Alaoui, A. E. (2018). The green dynamic vehicle routing problem in sustainable transport. In *2018 4th International Conference on Logistics Operations Management (GOL)*, pages 1–6. IEEE.
- Minocha, B., Tripathi, S., and Mohan, C. (2011). Solving vehicle routing and scheduling problems using hybrid genetic algorithm. In *2011 3rd international conference on electronics computer technology*, volume 2, pages 189–193. Ieee.

- Reimann, M., Doerner, K., and Hartl, R. F. (2003). Analyzing a unified ant system for the vrp and some of its variants. In *Workshops on Applications of Evolutionary Computation*, pages 300–310. Springer.
- Rizzoli, A. E., Oliverio, F., Montemanni, R., and Gambardella, L. M. (2004). Ant colony optimisation for vehicle routing problems: from theory to applications. *Galleria Rassegna Bimestrale Di Cultura*, 9(1):1–50.
- Taha, A., Hachimi, M., and Moudden, A. (2017). A discrete bat algorithm for the vehicle routing problem with time windows. In *2017 International Colloquium on Logistics and Supply Chain Management (LOGISTIQUA)*, pages 65–70. IEEE.
- Tan, K. C., Chew, Y. H., and Lee, L. H. (2006). A hybrid multiobjective evolutionary algorithm for solving vehicle routing problem with time windows. *Computational Optimization and Applications*, 34(1):115–151.
- Toth, P. and Vigo, D. (2001). The vehicle routing problem: Society for industrial and applied mathematics. *Siam Monographs on Discrete Mathematics and Applications*.
- Udin, K., Gui, R., and Rahmawan, A. (2017). Green vehicle routing problem with heterogeneous fleet and time windows. In *Proceedings of the 6th International Conference on Software and Computer Applications*, pages 223–227.
- Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., and Rei, W. (2012). A hybrid genetic algorithm for multi-depot and periodic vehicle routing problems. *Operations Research*, 60(3):611–624.
- Xiao, L., Hajjam-El-Hassani, A., and Dridi, M. (2017). An application of extended cuckoo search to vehicle routing problem. In *2017 International Colloquium on Logistics and Supply Chain Management (LOGISTIQUA)*, pages 31–35. IEEE.
- Yasin, M. and Vincent, F. Y. (2013). A simulated annealing heuristic for the green vehicle routing problem. In *Proceedings of the institute of industrial engineers Asian conference 2013*, pages 1261–1269. Springer.
- Zitzler, E. (2001). Spea2: Improving the performance of the strength pareto evolutionary algorithm. *Computer Engineering and Communication Networks Lab (TIK)*.
- Zitzler, E., Laumanns, M., and Bleuler, S. (2004). A tutorial on evolutionary multiobjective optimization. *Metaheuristics for multiobjective optimisation*, pages 3–37.