A Multi-objective Optimization Approach for the Capacitated Vehicle Routing Problem with Time Windows (CVRPTW)

Wissam Marrouche[1(✉)], Haidar M. Harmanani[2], and Janka Chlebíková[1]

[1] School of Computing, University of Portsmouth, Portsmouth, UK
{wissam.marrouche,janka.chlebikova}@port.ac.uk
[2] Computer Science Department, Lebanese American University, Byblos, Lebanon
haidar@lau.edu.lb

Abstract. Investigating complex combinatorial optimization problems such as the capacitated vehicle routing problem with time windows (CVRPTW) commonly benefits the advancement of logistic systems and telecommunications fields. CVRPTW is a elaborate strain of the well studied capacitated vehicle routing problem (CVRP) with a time window framework as an added constraint. As the problems are known to be NP-hard, meta-heuristics proficiently explore the search space to obtain near optimal solutions. In this manuscript, we explore the strength Pareto evolutionary algorithm SPEA2 hybridized with a hill climbing method as a local search, which yields a very sophisticated yet promising population based meta-heuristic algorithm. The novelty of our approach is in the selection of multi objective evolutionary algorithm namely SPEA2, its proposed evolutionary operators, and the optimization for three targets: the total distance traveled, the number of routes, and the average time per route - a newly introduced objective -, which offers an interesting trade-off opportunity scenarios for practitioners or supply chain managers. Based on our experimental results, such multiple objective optimization can guide the search toward favorable reduction of the total distance as well as offering a margin of compromise in the selection of the objectives of the final solution in general. The presented algorithm is a Python implementation tested extensively on the `Solomon's instances benchmark`. Encouraging results are recorded (sometimes reaching the best known in literature) and hints for further refinement are proposed for future research.

Keywords: Capacitated Vehicle routing problem with time windows · Strength pareto evolutionary algorithm · Hill climbing · Multi objective optimization · Evolutionary operators · Meta-heuristics

1 Introduction

The vehicle routing problem (VRP) is one of the most studied combinatorial optimization problem that arises in application logistic systems such as organizing delivery and/or collection of payloads in a geographical region. The advancement brought to

the problem can not only benefit to applied operational research or management science [27], but it has a major economic (and environmental) impact [1]. For example, in food and drink industry "added cost" of distribution can contribute up to 70% of intrinsic good cost [7].

The VRP problem [30] can be *stretched* out by imposing constraints on one or more variables of the standard problem. The capacitated vehicle routing problem (CVRP) is the standard VRP, where a limited vehicle capacity is taken into account along with routes optimization. If we relax the multiple route and vehicle capacity constraints, hence all customers must be served by a single route, the CVRP reduces into the *Travelling Salesman Problem*, an NP-hard problem [27].

In this paper we study the CVRP problem with time windows constraints, CVRPTW, similarly to [30]. In this paper, we expand on the work in [22] and consider multi-identical vehicles and single depot while optimizing for some specific objectives that we will describe next. However this work can be extended to multi-depot by partitioning the customers to zones and running the CVRPTW algorithm version in parallel [9].

Exact methods like integer linear programming (ILP) are very computationally demanding to be even considered to solve for the NP-hard CVRPTW problem [6]. Moreover, the time windows introduces a series of precedence on visits, which turns the problem asymmetric, despite the fact that the distance and time matrices were initially symmetric. In practice, heuristics and meta-heuristic methods can be used in order to achieve near-optimal solutions within reasonable amount of computation time. Such approaches provide good solutions for industrial scale problems [27]. We approach the CVRPTW problem utilizing the population-based strength Pareto evolutionary algorithm (SPEA2) as conceived by Zitzler et al. [36] but augmented - or hybridized - - with a hill climbing local search to guide the search toward favorable solutions. More precisely, this work experiments with different genetic operators types such as uniform/fixed point recombination operators and their hybrid combination with other operators, where the genetic operators are followed by hill climbing operator in order to improve the final quality of the solution. Therefore, we also tuned the parameter setting of the proposed operators to reach satisfactory solutions. The advantage of our approach is that it ameliorates the search space toward good solutions by optimizing for three main objectives namely the total distance, the number of routes (equal to the number of vehicles deployed) and the average time per route which is an objective primarily needed to counterbalance the distinct vehicles operation time and route lengths. As a consequence, the proposed algorithmic approach applied to well entrenched CVRPTW benchmarks – and compared to a relatively close existing one – offers a myriad of Pareto optimal/sub-optimal diverse solutions and thus allowing a trade-off analysis in the selection of a desired solution that matches the implementation needs of the domain expert practitioners or supply chain managers.

The remainder of the paper unfolds as follows: Sect. 2 provides a formal introduction of the CVRPTW problem, Sect. 3 lists concisely the VRP related work found in the literature and Sect. 4 gives a comprehensive synopsis of the methodology of our brought forward approach. Section 5 presents and analyzes the simulation results. Lastly, Sect. 6 summarizes the findings and provides suggestions for further research directions.

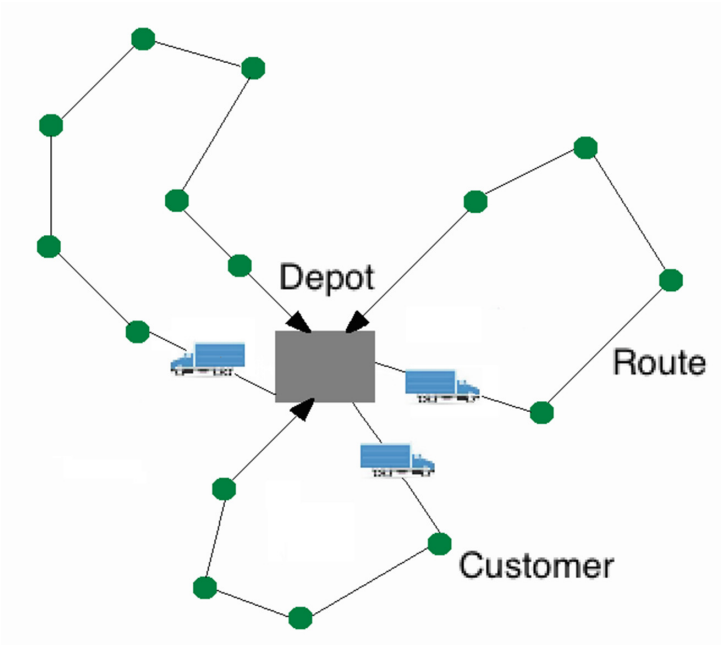

Fig. 1. Typical representative solution of VRP [22].

2 Formal Model

Our formal definition of the CVRTPW problem is based on the one from [16]. 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 customers 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. Each out of K vehicles is related to one route in the network and has the same capacity q_k and the demand m_i of every customer must be serviced only one time by one vehicle. The aggregation of the capacity of all the demands on a given route can't exceed q_k capacity of vehicle k embarking on this route.

The time window restrictions for each customer 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 must stand by. Moreover, a service time f_i for each customer for a given loading and unloading time is taken into consideration. Each vehicle is expected to complete its route within the total route time r_k.

There are further simplifications in the model, as 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. Also loading and unloading time to be unique regardless of the amount of the demands.

This is formalized by the following subsequent parameters and variables:

- $x_{ijk} = \begin{cases} 1 \text{ if vehicle } k \text{ travels from customer } i \text{to customer } j \\ 0 \text{ otherwise} \end{cases}$
  $i \neq j; i, j \in \{0, 1, 2, \ldots N\}; k \in \{1, \ldots, K\}$
- 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
- t_i arrival time at customer i
- w_i waiting time at customer i for the service operation to be ready to start
- r_k maximum route time for vehicle k

Based on these parameters, the goal of the CVRPTW problem is to minimise the objective function

$$\min \sum_{i=0}^{N} \sum_{j=0, j\neq i}^{N} \sum_{k=1}^{K} c_{ij} x_{ijk} \tag{1}$$

with subject to the following constraints:

$$\sum_{k=1}^{K} \sum_{j=1}^{N} x_{0jk} \leq K \quad \text{(maximum } K \text{ vehicles leave the depot)} \tag{2}$$

$$\sum_{j=1}^{N} x_{0jk} = \sum_{j=1}^{N} x_{j0k} \leq 1 \quad \text{for } k \in \{1, 2, \ldots, K\} \tag{3}$$

(every route starts and terminates at the depot)

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

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

(4 and 5 ensure that each customer is served by exactly one vehicle)

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

(capacity of each vehicle can't be exceeded)

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

(travel time constraints for each vehicle can't be exceeded)

$$t_0 = w_0 = f_0 = 0 \tag{8}$$

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

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

(constraints (8), (9), and (10) are related to the times window)

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

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.

All these parameters are considered in `Solomon`'s 1987 model, and `Solomon`'s 56 100-customers instances of CVRPTW model that we'll be using for comparison of our results.

In this paper we look on the correlation of three objectives: total distance, number of routes, and average time per route concurrently in attempt to find the best solution for the single depot CVRPTW in a reasonable time. Reducing the average time per route is an important aspect to consider from practical point of view as it decreases the driver cost time and balances the routes lengths.

3 Various Approaches to the Problem

Heuristics such as simulated annealing and its improvements [4,35], together with an iterated local search [35] indicate that one can get 'sufficiently good' solutions of the CVRPTW problem within an reasonable amount of time. This approach is based on a limited exploration of the search space quickly, while meta-heuristics abide by the principles of intensification and diversification in their search [27]. Population based meta-heuristic algorithms work well for global searches offering global exploration and local exploitation capability over simple heuristics. In this context, the population-based approach with several evolutionary stages and swarm intelligence-based algorithms like particle swarm optimization, ant colony optimization, artificial bee colony algorithm can outperform simple heuristics when solving VRPTW [8]. Another bio-inspired meta-heuristics applied to the basic CVRP problem is 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 [34]. Cuckoo search solves efficiently even more elaborate VRP problems such as the VRP with heterogeneous fleet,

mixed backhauls (pickup and delivery) and time windows [23]. 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 [28]. There are also other bio-inspired approaches considered in literature such as artificial immune systems and neural networks. For the VRP problem many contributions have been made using hybrid algorithms: using a greedy method for an initial solution followed by tabu search [31] or based on ant colony optimization algorithm [27] with a large neighborhood search (LNS) algorithm [24]. In general, meta-heuristics have the tendency to yield good results when hybridized with local search methods [8,27], adding specific problem intelligence in the enhancement of the solutions. A hybrid genetic algorithm was proposed in [32] for multi-depot and periodic VRP. A similar published work to our studied problem can be found in [3] and [14], which implements multi objective vehicle routing problem with time windows and optimizes for two concurrent objectives, namely the total distance and the number of vehicles to obtain the set of Pareto optimal solutions. In [14], enhanced reproduction operators are proposed for the CVRPTW problem. An even closer work to this paper is found in [29], which implements a hybrid multi objective evolutionary algorithm (MOEA) accommodating the sequence-oriented optimization. Also [11] propose an improved MOEA for the VRPTW optimising mainly for two objectives: the number of routes and the total distance traveled. 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 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 traveled and therefore ensure a fast convergence. In fact, [36] 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 [17]. 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. This paper also proposes its own evolutionary operators similarly to the another paper [13] where tailored intelligent neighborhood moves to the genetic operators and guides the search toward optimal results irrespective of the quality of the initial solution. Promising results were achieved using the new approach which makes it a good reference for comparison in the presence of three objectives.

4 Our Approach

Variants of local search are the preferred heuristics for otherwise complex problems such as graph partitioning and the vehicle routing [12]. Our proposed methodology tackling the CVPRTW problem is based on the hybrid SPEA2-local search algorithm as detailed in the following subsections:

4.1 Setting the Input Parameters

Based on the extracted information from the `Solomon`'s benchmark 100-customer 56 instances (1987) (see [8] for more details), we populate two data structures namely: Nodes list $Nodes$, and the graph matrix G about the coordinates (i_x, i_y) of the customer i to retrieve the cost $c_{i,j}$. Each node in $Nodes$ contains information related to the customer i: demand m_i, earliest arrival time e_i (opening time), service time f_i and due date of the delivery l_i (closing time). The scheduling has to enforce that the arrival time t_i to a customer i lies within the time window $[e_i, l_i]$. As for the arrival time t_i, the waiting time w_i for the service operation to be opened, and the travel time between two nodes $t_{i,j}$ are computed dynamically during the scheduling process.

A typical solution S to our studied problem is represented as an object with the following fields: an array of routes where each $route_k$ contains the sequence of visited customers, an average time per route, and a total traveled distance. In fact, the minimization of average completion time per route is linked to work-in-progress inventory management [18] 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 we visit the nodes in a different random order). The algorithm loops to visit all nodes - in the random order - that are not previously assigned to any given route until all nodes are assigned to routes. The starting solution has an empty list of routes. In each iteration we create a new route and add to it all customers i, ($i \in \{1 \ldots, N\}$) that are not allocated to any route so far keeping all the constraints valid. Our proposed greedy algorithm ensures that the obtained greedy solution meet the constraints: the vehicle capacity constraint and the time windows restrictions. A validate function is implemented to ensure that the solution is feasible i.e. meets the aforementioned constraints. Here, the validate function does not check if the total number of allocated vehicle surpasses the fleet size for a given greedy solution and thus considers the generated greedy solution as valid. The check for the number of used vehicles is postponed till the end of the overall simulation as in the process of improvement of the initial solution, the number of vehicles may decrease. If at the end of the simulation a solution with a fleet size violation is obtained then such solution will be discarded. In our `Solomon's` benchmarks simulations, the fleet size violation never occurred as we see in the presentation of our results in Sect. 5.

We generate a random pool of initial solutions relying on the previously defined greedy algorithm. The challenge is now to advance the search starting from the obtained inception state in a proficient meta-heuristic approach that we base on SPEA2 hybridized with local search. In the following we define a few evolution operators that are building blocks for the overall approach that will be described in details.

4.3 Tweak Operator

The *tweak operator* produces each time a slightly randomly different candidate solution. It is also called a modification procedure that makes search exploitative by exchanging two customers position either inter route insertion swap or intra routes insertion exchange.

More precisely, there are two options for the tweak operators, let S be an initial solution: (a) two randomly selected nodes in S are within the same route – in that case we just swap two nodes inside the route; (b) two randomly selected nodes i and j are in different routes namely the route k_i and route k_j. In this case we firstly move the node i from the route k_i to the route k_j and insert the node i after the node j in the route k_j. This modification S to S^* can decrease the number of routes as after applying it, the route k_i might become empty. In such case we simply remove it from S^*. Using the validate operator, we check a feasibility of the solution: if it is valid we return S^*; otherwise we roll back to the initial solution S. Note that S^* may not be necessarily better than S. We repeat the tweak operation and if no valid S^* is found after a fixed number of trials, we return the initial solution S. In our implementation, we have repeated the trials maximum 9 times.

4.4 Recombination Operator

The *recombination operator* is an "asexual recombination" as illustrated in Fig. 2. One of its advantages is that it explores the search space and avoids the fall into a local minima. The implementation is given as follows:

Given an initial solution S, we select randomly two different routes –or paths– in the solution and perform a fixed-point recombination from the middle of each route, which results in two new offspring routes that can be optionally merged together using a uniform combination. The resulting solution, S^*, is verified for feasibility using the validate operator. If the solution is not feasible then the operator is repeated for a number of times that is equal to the number of routes in S plus a bias value that we tune empirically. At any moment the trial can terminate if S^* is feasible. Otherwise, the solution S is kept.

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}

Fig. 2. Recombination Example: Fixed Point and Uniform [22].

4.5 Fuse Operator: Naïve Merge

The *fuse operator* is also an asexual recombination merging two randomly selected routes of a given solution by naïvely queuing one route after the other to create one path or one offspring route that will be added to the list of routes while removing the two selected parent routes from the solution. However, this move is only efficacious

at the beginning of the meta-heuristic deployment. In fact, at early stages it is highly probable that there exists a room for improvement over the initial pool of solutions. However in general, this operator will not improve the existing solution at later stages. This is the reason why we only apply the operator with a low probability in order not to slow down the simulation without improvement. Its implementation details can be described as follows:

Given an initial solution S, we create a new solution S^* in which two random parent routes in S are joined to form a new route: the last non-depot node of the first randomly selected route is connected with the first non-depot vertex of the second randomly selected route and the two parents random routes are moved from S^*. The solution S^* is checked using the validate operator; if it is a valid one, we return S^*. Otherwise we rollback and repeat the trial a number of times equal to the initial total number of routes in S plus a bias or we stop earlier, if generated S^* is valid. If the trial fails, we return the initial solution S.

4.6 Tuning Operator

The *tuning operator* is applied after the genetic operators and helps in performing a local search in order to converge towards an optimal solution or in the worst case return the same initial solution if no improvement is obtained. In fact, the steepest ascent hill climbing with replacement operator [20] is used to reduce the total cost (distance traveled) of a given offspring solution; in other words, to force the evolution of population downhill for this objective. Thus, each time the tuning operator is invoked, it performs either a tweak or a recombination (mutually exclusive) operation with a certain probability to be determined empirically. The rational behind this can be justified by the fact that we need to switch back and forth from one type of operator to another in order to escape local minima, see Fig. 3.

Steepest Ascent Hill Climbing

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

Fig. 3. Tuning Operator as a Local Search.

4.7 SPEA2 Fitness Computation and Archive Construction

Our approach to attack the CVRPTW problem is based on the three objectives functions that need to be minimized concurrently, namely the total distance of routes D which impacts the fuel cost, the number of routes K which impact the number of vehicles needed, and the average time per route T_{avg} which impacts the driver's working time.

Formally, and based on Eq. 1 the total distance D is given as in Eq. 12:

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

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

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

In general, the *fitness function* is very important as it helps guiding the algorithm towards the local optima [36]. In our case, the fitness function is a vector-valued function that rely on the Pareto dominance concept and is defined as:

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

Following the definition from [37], 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.

To each ach individual solution A_i in the archive A (as introduced in Fig. 4) as well as P_i in population P (refer to Fig. 5) is given a strength value, $Strength(i)$, denoting the number of solutions it dominates as described by Eq. 15.

$$Strength(i) = \sum_{i,j \in P \cup A} [i \succ j], \tag{15}$$

where the bracked notation $[i \succ j]$ is 1 if $i \succ j$ is true and 0 otherwise. Based on this definition, the raw fitness $f(i)$ of an individual solution i is determined as follows [21]:

$$f(i) = \sum_{i,j \in P \cup A, j \succ i} Strength(j). \tag{16}$$

The raw fitness is evaluated by the strengths of its denominators 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. Possible ties are resolved using a nearest neighbor density estimation function, $Density(i)$.

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

where σ_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 $O(M_P^2 \log M_P)$ as described in [21].

SPEA2 Archive

$P \leftarrow \{P_1, \ldots, P_m\}$ ▷ population
$O \leftarrow \{O_1, \ldots, O_n\}$ ▷ objective to assess with
$a \leftarrow$ desired archive size
$A \leftarrow$ Pareto non-dominated front of P ▷ the archive
$Q \leftarrow P$ - A
if $|A| < a$ **then**
    Sort Q by fitness
    $A \leftarrow A \cup$ the $a - |A|$ fittest individuals in Q, breaking ties arbitrarily
while $|A| > a$ **do**
    $Closest \leftarrow A_1$
    $c \leftarrow$ index of A_1 in P
    for all individual $A_i \in A$ except A_1 **do**
        $l \leftarrow$ index of A_i in P
        for k = 1 to m-1 **do**
            if DistKthNearest(k, P_l) < DistKthNearest(k, P_c) **then**
                $Closest \leftarrow A_i$
                $c \leftarrow l$
                break
            else
                if DistKthNearest(k, P_l) > DistKthNearest(k, P_c) **then**
                    break
        $A \leftarrow \{Closest\}$
return A

Fig. 4. SPEA2 Archive Construction Algorithm [20].

4.8 SPEA2 Algorithm

The *strength pareto evolutionary algorithm* (SPEA2) is a multiple objective evolutionary optimization algorithm that localizes and preserves a front of non-dominated solutions, preferably a set of Pareto optimal solutions. We achieve this goal by constructing an archive as in Fig. 4 and relying on an evolutionary routine as introduced in Sect. 4.9. We propose the deployment of the following SPEA2 algorithm pseudo code in Fig. 5 based on [20].

4.9 Evolve Operator

SPEA2 requires an evolution mechanism, which we base on the following four operators: tweak, recombine, fuse, and tuning. The algorithm applies these operators by taking turn, and using probabilities that are determined empirically (Sect. 5.1). Together,

SPEA2 Algorithm

$m \leftarrow$ desired population size
$a \leftarrow$ desired archive size ▷ typically a = m
$P \leftarrow \{P_1, \ldots, P_m\}$ ▷ build initial population using a greedy method Section 4.2
repeat
    AssessFitness(P)
    $P \leftarrow P \cup A$
    $BestFront \leftarrow$ Pareto Front of P
    $A \leftarrow$ Construct SPEA2 Archive of size a from P
    $P \leftarrow$ Evolve(A), using tournament selection of size 3
until number of generations reached
return $BestFront$

Fig. 5. SPEA2 Algorithm Pseudo Code based on [20].

the diverse and comprehensive set of operators exploit and explore the search space while escaping the local minima.

5 Summary of Our Experimental Results

5.1 Hyperparameters Tuning

Evolutionary algorithms are stochastic in nature and thus very sensitive to hyperparametres. Therefore, one of the main tasks was tuning the problem's parameters. Thus, fixed-point and uniform recombination and we selected the better of the two reported solutions. The experiment was repeated 18 times for each recombination type (fixed point and uniform) and resulted with the identification of two main configurations groups. Table 2 shows the problem's parameters such as the bias is not null which makes the search operators run more iteration and thus improve solutions through intensive neighborhood search (i.e. more exploitative iterative improvement for local searches at the expense of execution time of the whole meta-heuristic solution) which justifies reducing number of generation for our meta-heuristic setting. Where as Table 1 is more exploratory i.e. the number of generation parameter is higher since a bias of zero implies less number of iterations for the deterministic operators, thus pushing our method to aggressively explore the search space from a higher-level framework perspective.

Exceptionally, for the R1 benchmark some obtained results were better using the first configuration while for others the second configuration showed better results, 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 1. Parameters settings of SPEA2 for C1, RC1 benchmarks [22].

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

Table 2. Parameters settings of SPEA2 for C2, RC2 benchmarks [22].

Parameter	# Gen	Tweak	Recombine	Fuse	popsize	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 [22].

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

continued

Table 3. continued

Benchmark	Ours			BCO	
	Distance	Routes	Avg Time	Distance	Routes
RC108	**1156.92**	11	212.29	1165.17	11
RC201	1345.36	6 ▽	748.61	**1315.57**	7
RC202	1174.23	6 ▽	716.13	**1169.72**	5
RC203	1038.97	5 ▽	777.83	**1010.74**	4
RC204	**883.45**	4 ▽	712.01	890.28	4
RC205	1237.81	7 ▽	721.89	**1221.28**	7
RC206	1147.15	6 ▽	702.32	**1097.65**	6
RC207	1061.85	5 ▽	672.37	**1024.17**	5
RC208	865.07	5 ▽	612.30	**864.56**	4

[a] We list our best solution among the recombination methods (fixed and uniform)
[b] ▽ a solution exists where the number of routes is lower than the reported given a larger total distance

5.2 Results Analysis

We implemented the proposed hybrid SPEA2 algorithm using Python and experimented with various benchmarks on an Intel Xeon CPU E5-2650 with 16 cores. We attempted the `Solomon's benchmark` and received favorable results as shown in Table 3. For each benchmark, we report the solution with the best Pareto optimal answer belonging to one of the 36 Pareto fronts in all 36 runs. Although there are results in literature with lower routes cost, none of the reported solutions optimized for these three objectives: distance, average time per route, and number of vehicles simultaneously.

For each instance, one run took on average around one day, which is not unexpected given the problem's intractability and the complexity of dealing with optimizing three constraints. Thus, it is difficult to compare to reported works. Nonetheless, to have a meaningful reference to compare achieved results, we matched our results with the two objective based CVRPTW problem. In fact, if we compare our solution with the absolute best reported results for two objectives in literature [2] and [14], we obtain either optimal or sub optimal solutions. But for more practical matter we compared our results to the bee colony optimization (BCO) algorithm introduced by [16] 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 better results than the second configuration mainly in case when the number of expected routes is high (based on BCO algorithm findings as a reference). Some exceptions were observed where the second configuration gave better results for given benchmarks of type R1 (refer to Table 3, parameter type */+ for a given R1 benchmark).

Finally, it should be noted that while executing all the simulations, we 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. 6. 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. 7. This is in complete match with the literature [25].

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

Fig. 6. 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

Fig. 7. RC202 Typical Solution.

Figure 8, Fig. 9, Fig. 10 and Fig. 11 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 distance variation at each run (18 times i.e. 18 Pareto sets) of a given benchmark (given uniform or recombination operator option). Thus, the experiment is repeated 36 times (2×18 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 in the distribution and skewness is noticed between fixed and uniform recombination options.

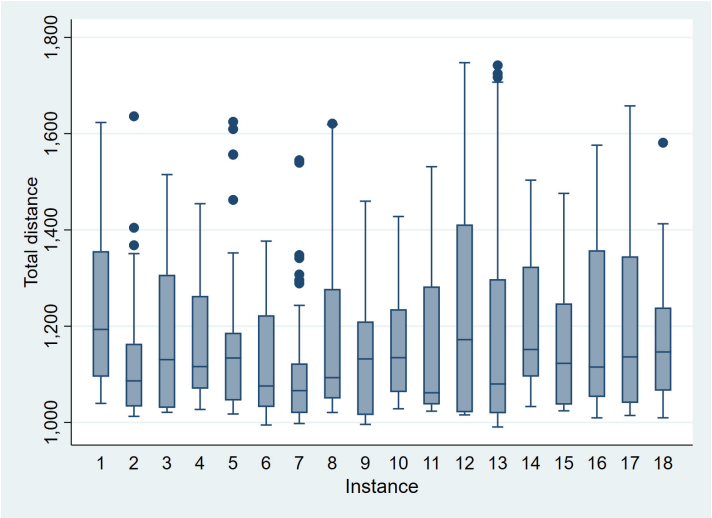

Fig. 8. Variance analysis for Solomon_100 R108 with a uniform recombination operator [22].

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. 12, 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

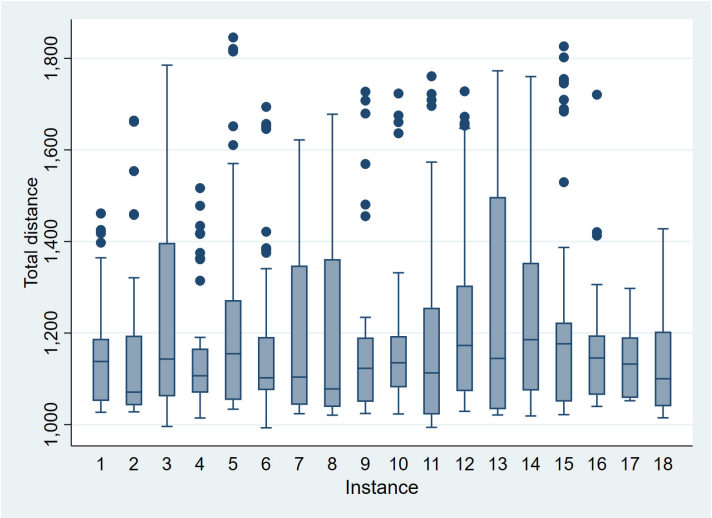

Fig. 9. Variance analysis for Solomon_100 R108 with a fixed recombination operator [22].

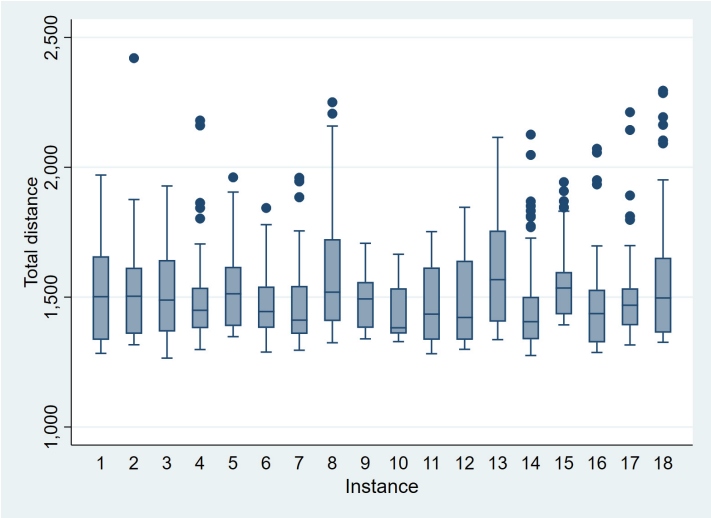

Fig. 10. Variance analysis for Solomon_100 R201 with a uniform recombination operator [22].

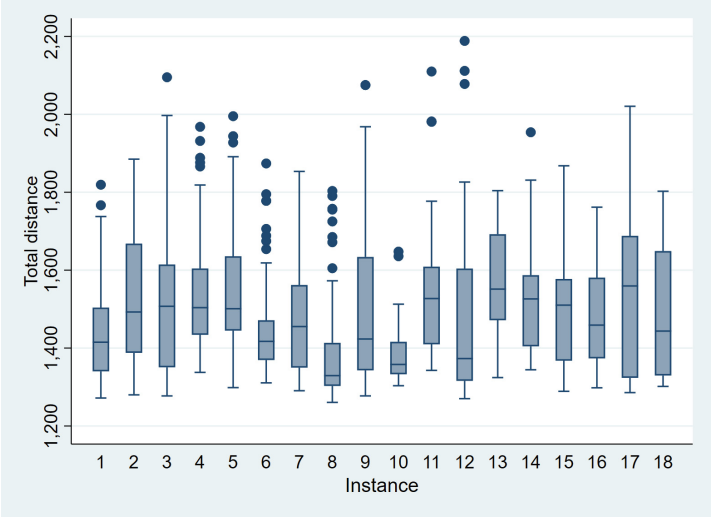

Fig. 11. Variance analysis for Solomon_100 R201 with a fixed recombination operator [22].

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 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. 12, a good value of $HV = 0.85$ is obtained given a reference point $r = 1.3$ (30% larger than the nadir point) [15].

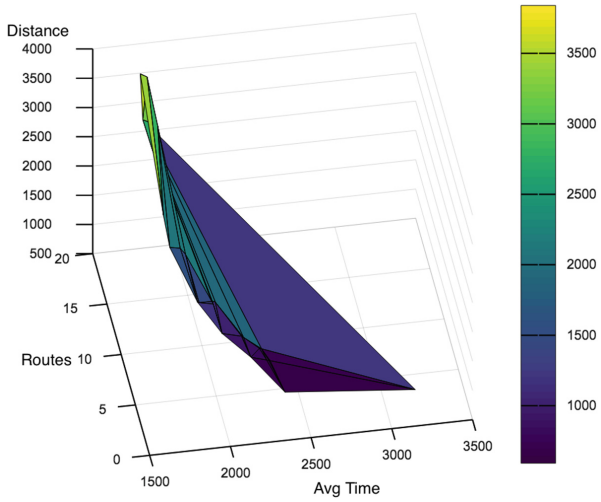

Fig. 12. A visual representation sample of the Pareto front surface obtained for Solomon_100 C201 [22].

Lastly, the visualization of the obtained solutions is very helpful in presenting the final results. In this context, Fig. 13, Fig. 14, and Fig. 15 are topological maps of sample benchmarks (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 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 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 Conclusions

In the logistic realm, it is incredibly hard task to determine and integrate the most optimized routing schedule given the different imposed constraints or objectives that might even be conflicting each other in some cases. To find a satisfyingly good solution for such an optimization problem, the meta-heuristics seem to fill in this trad-off gap. The

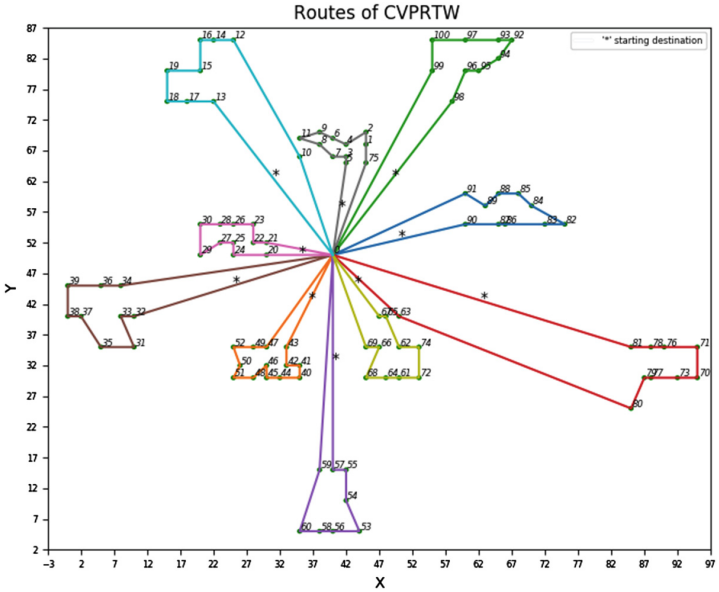

Fig. 13. Network topology for Solomon_100 C104: distance = 824.78, routes = 10, time = 995.40 [22].

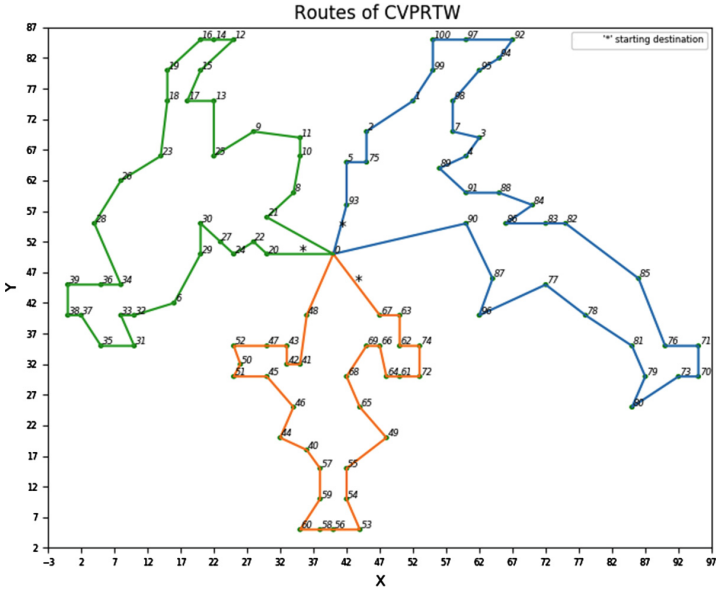

Fig. 14. Network topology for Solomon_100 C201: distance = 591.56, routes = 3, time = 3197.19 [22].

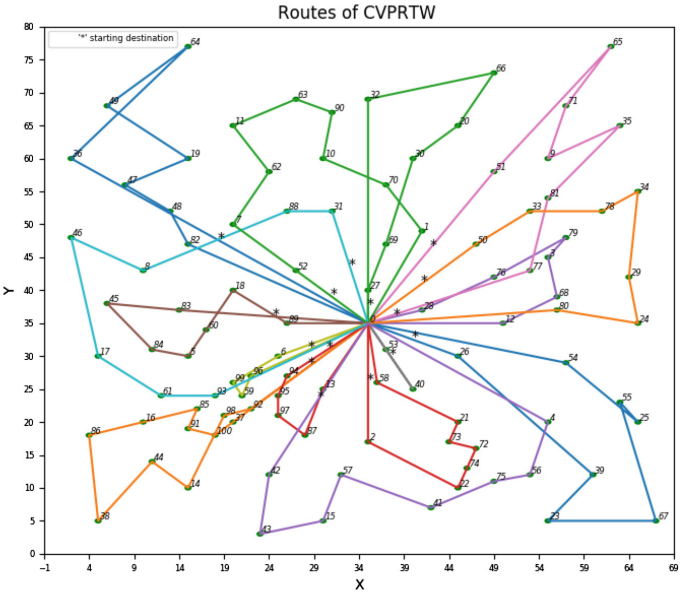

Fig. 15. Network topology for Solomon_100 R103: distance = 1240.08, routes = 15, time = 196.46 [22].

VRP problem became the major problem of E-commerce retailers and its significance even increased during the recent pandemic which amplifies the importance of our contribution to a proposed solution.

In this work, we propose a novel meta-heuristic approach using a hybrid SPEA2 augmented with a hill climbing as a local search tuning operator to solve for the CVRPTW problem by optimizing for three objectives: the total distance traveled, the number of routes and the average time per route. The last objective is well motivating in order 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 [26]. 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 adapt dynamically to changing objectives or deploying it on different variants of VRP such as: the green version of the CVRPTW that accounts for carbon emissions [19], or

VRP with soft timing constraints, which if violated must account for a penalty. Ambitious future work can also account for traffic, i.e. higher cost routes in city centers [27] in what is known as time dependent VRPTW. Another extension could be to target split delivery vehicle routing problem with time windows 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 [5,10].

On the other hand, this work can be generalized to address a different class of VRP altogether, namely the VRP with backhauls (pick up and delivery) [30]. Moreover, in some "rich VRP scenarios" it is not necessary to visit all customers, and the distributor chooses to services its customers relying on their attractiveness (either multiple times visit or with multiple planning days) [33]. Also, the proposed meta-heuristic solution can be a building block to a parallel implementation version with different partitioned zones (i.e. multi-depot solution) and thus service a very large number of customers proficiently.

Acknowledgements. We thank dearly Dr. Mohamad Bader-El-Den from the School of Computing at University of Portsmouth for his valuable comments. 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

1. Berger, J., Barkaoui, M., Bräysy, O.: A route-directed hybrid genetic approach for the vehicle routing problem with time windows. INFOR Inf. Syst. Oper. Res. **41**(2), 179–194 (2003)
2. Bychkov, I., Batsyn, M.: A hybrid approach for the capacitated vehicle routing problem with time windows. In: OPTA-SCL. CEUR Workshop Proceedings (2018)
3. Chiang, T.C., Hsu, W.H.: A knowledge-based evolutionary algorithm for the multiobjective vehicle routing problem with time windows. Comput. Oper. Res. **45**, 25–37 (2014)
4. Czech, Z.J., Czarnas, P.: Parallel simulated annealing for the vehicle routing problem with time windows. In: Proceedings 10th Euromicro Workshop on Parallel, Distributed and Network-Based Processing, pp. 376–383. IEEE (2002)
5. Da Silva, M.D.M., Subramanian, A., Ochi, L.S.: 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 (2014)
6. Dash, M., Anekal, B.I.: Variables reduction in vehicle routing problems. SPC ERA IJBM **2**(6) (2014)
7. De Backer, B., Furnon, V., Prosser, P., Kilby, P., Shaw, P.: Local search in constraint programming: application to the vehicle routing problem. In: Proceedings CP-97 Workshop Industrial Constraint-Directed Scheduling, pp. 1–15. Schloss Hagenberg Austria (1997)
8. Dixit, A., Mishra, A., Shukla, A.: Vehicle routing problem with time windows using meta-heuristic algorithms: a survey. In: Yadav, N., Yadav, A., Bansal, J.C., Deep, K., Kim, J.H. (eds.) Harmony Search and Nature Inspired Optimization Algorithms. AISC, vol. 741, pp. 539–546. Springer, Singapore (2019). https://doi.org/10.1007/978-981-13-0761-4_52
9. Dumez, D., Lehuédé, F., Péton, O.: A large neighborhood search approach to the vehicle routing problem with delivery options. Transp. Res. Part B Methodol. **144**, 103–132 (2021)
10. Feillet, D., Dejax, P., Gendreau, M., Gueguen, C.: Vehicle routing with time windows and split deliveries. Technical Paper 851 (2006)

11. Garcia-Najera, A., Bullinaria, J.A.: An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows. Comput. Oper. Res. **38**(1), 287–300 (2011)
12. Ghosh, D., Chakravarti, N.: A competitive local search heuristic for the subset sum problem. Comput. Oper. Res. **26**(3), 271–279 (1999)
13. Hosny, M.I.: Investigating heuristic and meta-heuristic algorithms for solving pickup and delivery problems. Cardiff University (United Kingdom) (2010)
14. Hsu, W.H., Chiang, T.C.: A multiobjective evolutionary algorithm with enhanced reproduction operators for the vehicle routing problem with time windows. In: 2012 IEEE Congress on Evolutionary Computation, pp. 1–8. IEEE (2012)
15. Ishibuchi, H., Imada, R., Setoguchi, Y., Nojima, Y.: How to specify a reference point in hypervolume calculation for fair performance comparison. Evol. Comput. **26**(3), 411–440 (2018)
16. Jawarneh, S., Abdullah, S.: Sequential insertion heuristic with adaptive bee colony optimisation algorithm for vehicle routing problem with time windows. PLoS ONE **10**(7), e0130224 (2015)
17. King, R.A., Deb, K., Rughooputh, H.: Comparison of NSGA-II and SPEA2 on the multiobjective environmental/economic dispatch problem. Univ. Mauritius Res. J. **16**(1), 485–511 (2010)
18. Li, W., Dai, H., Zhang, D.: The relationship between maximum completion time and total completion time in flowshop production. Procedia Manufact. **1**, 146–156 (2015)
19. Lin, C., Choy, K.L., Ho, G.T., Chung, S.H., Lam, H.: Survey of green vehicle routing problem: past and future trends. Expert Syst. Appl. **41**(4), 1118–1138 (2014)
20. Luke, S.: Essentials of metaheuristics. lulu, 2013. Metaheuristics in large-scale global continues optimization: a survey (2013). http://cs.gmu.edu/sean/book/metaheuristics
21. Marrouche, W., Farah, R., Harmanani, H.M.: A strength pareto evolutionary algorithm for optimizing system-on-chip test schedules. Int. J. Comput. Intell. Appl. **17**(02), 1850010 (2018)
22. Marrouche, W., Harmanani, H.: A strength pareto evolutionary algorithm for solving the capacitated vehicle routing problem with time windows. In: Proceedings of the 13th International Joint Conference on Computational Intelligence (IJCCI 2021), pp. 96–106 (2021)
23. Meryem, B., Abdelmadjid, B.: Resolving a vehicle routing problem with heterogeneous fleet, mixed backhauls and time windows using cuckoo behaviour approach. Int. J. Oper. Res. **24**(2), 132–144 (2015)
24. Messaoud, E., El Idrissi, A.E.B., Alaoui, A.E.: The green dynamic vehicle routing problem in sustainable transport. In: 2018 4th International Conference on Logistics Operations Management (GOL), pp. 1–6. IEEE (2018)
25. Minocha, B., Tripathi, S., Mohan, C.: Solving vehicle routing and scheduling problems using hybrid genetic algorithm. In: 2011 3rd International Conference on Electronics Computer Technology, vol. 2, pp. 189–193. IEEE (2011)
26. Reimann, M., Doerner, K., Hartl, R.F.: Analyzing a unified ant system for the VRP and some of its variants. In: Cagnoni, S., et al. (eds.) EvoWorkshops 2003. LNCS, vol. 2611, pp. 300–310. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36605-9_28
27. Rizzoli, A.E., Oliverio, F., Montemanni, R., Gambardella, L.M.: Ant colony optimisation for vehicle routing problems: from theory to applications. Galleria Rassegna Bimestrale Di Cultura **9**(1), 1–50 (2004)
28. Taha, A., Hachimi, M., Moudden, A.: A discrete bat algorithm for the vehicle routing problem with time windows. In: 2017 International Colloquium on Logistics and Supply Chain Management (LOGISTIQUA), pp. 65–70. IEEE (2017)
29. Tan, K.C., Chew, Y.H., Lee, L.H.: A hybrid multiobjective evolutionary algorithm for solving vehicle routing problem with time windows. Comput. Optim. Appl. **34**(1), 115–151 (2006)

30. Toth, P., Vigo, D.: The vehicle routing problem: society for industrial and applied mathematics. Siam Monographs on Discrete Mathematics and Applications (2001)
31. Udin, K., Gui, R., Rahmawan, A.: Green vehicle routing problem with heterogeneous fleet and time windows. In: Proceedings of the 6th International Conference on Software and Computer Applications, pp. 223–227 (2017)
32. Vidal, T., Crainic, T.G., Gendreau, M., Lahrichi, N., Rei, W.: A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. Oper. Res. **60**(3), 611–624 (2012)
33. Wen, M., Clausen, J., Larsen, J.: Rich vehicle routing problems and applications. DTU Management (2010)
34. Xiao, L., Hajjam-El-Hassani, A., Dridi, M.: An application of extended cuckoo search to vehicle routing problem. In: 2017 International Colloquium on Logistics and Supply Chain Management (LOGISTIQUA), pp. 31–35. IEEE (2017)
35. Yasin, M., Vincent, F.Y.: A simulated annealing heuristic for the green vehicle routing problem. In: Lin, Y.K., Tsao, Y.C., Lin, S.W. (eds.) Proceedings of the Institute of Industrial Engineers Asian Conference 2013, pp. 1261–1269. Springer, Singapore (2013). https://doi.org/10.1007/978-981-4451-98-7_149
36. Zitzler, E.: SPEA2: improving the performance of the strength pareto evolutionary algorithm. Computer Engineering and Communication Networks Lab (TIK) (2001)
37. Zitzler, E., Laumanns, M., Bleuler, S.: A tutorial on evolutionary multiobjective optimization. In: Metaheuristics for Multiobjective Optimisation, pp. 3–37 (2004)