The cumulative capacitated vehicle routing problem with min-sum and min-max objectives: An effective hybridisation of adaptive variable neighbourhood search and large neighbourhood search

Jeeu Fong Sze*, Said Salhi, Niaz Wassan

Centre of Logistics and Heuristics Optimisation (CLHO), Kent Business School, University of Kent, Canterbury, CT2 7PE, United Kingdom

Abstract

The cumulative capacitated vehicle routing problem (CCVRP) is a relatively new variant of the classical capacitated vehicle routing problem in which the objective is to minimise the sum of arrival times at customers (min-sum) instead of the total route distance. While the literature for the CCVRP is scarce, this problem has useful applications especially in the area of supplying humanitarian aid after a natural disaster. In this paper, a two-stage adaptive variable neighbourhood search (AVNS) algorithm that incorporates large neighbourhood search (LNS) as a diversification strategy is proposed. When tested on the benchmark data sets, the results show that the proposed AVNS is highly competitive in producing new best known solutions to more than half of the instances. An alternative but related objective that minimises the maximum arrival time (min-max) is also explored in this study demonstrating the flexibility and the effectiveness of the proposed metaheuristic. To the best of our knowledge, this is the first study that exploits the min-max objective of the CCVRP in addition to providing extensive computational results for a large number of instances for the min-sum. As a by-product of this study, managerial insights for decision making are also presented.

Keywords: Metaheuristics, cumulative capacitated vehicle routing, OR in disaster relief, min-max, CCVRP managerial insights, adaptive variable neighbourhood search.

1. Introduction

Since its establishment in more than fifty years ago, the classical vehicle routing problem (VRP) has been one of the most extensively studied topics within the area of transportation optimisation. Recently, many efforts have been taken to incorporate various attributes of real-life applications. One of these is by considering alternative objective functions such as minimising the total arrival

*Corresponding author

Email addresses: J.F.Sze@kent.ac.uk (Jeeu Fong Sze), S.Salhi@kent.ac.uk (Said Salhi), N.A.Wassan@kent.ac.uk (Niaz Wassan)

times of the vehicles, length of the longest route, the effect of pollution (Koç et al., 2016) and so on. The VRP which takes into account practical considerations is also termed the rich VRP (Hartl et al., 2006; Goel and Gruhn, 2008; Lahyani et al., 2015) or multi-attribute VRP (Vidal et al., 2014).

This study investigates a relatively new variant of the classical VRP namely the cumulative capacitated VRP (CCVRP). Here, the objective function is to minimise the sum of arrival times (min-sum) at the customers, subject to the vehicle capacity constraint. The CCVRP is motivated by the need of servicing the customers as soon as possible. This problem has many real-world applications such as in humanitarian logistics for transferring people to a safer place after a natural disaster (e.g., tsunamis or earthquakes), delivering food, medical supplies and necessities to affected regions and dispatching ambulances to patients' locations (Campbell et al., 2008). Other applications of the CCVRP arise in the school bus routing context where the total passengers' travel time is concerned (Bowerman et al., 1995) and in the home-delivery service (Méndez-Díaz et al., 2008) where the aim is to prioritize customers' satisfaction.

In the classical VRP, the routes are designed by optimising the total route length. In such case, some commodities may be served significantly later than others to minimise the total cost. To better reflect priorities and to ensure equity and fairness, a customer-centric objective function is usually used. For instance in the CCVRP, the waiting time of a service system from the customer's point of view is considered. This type of objective function can be viewed as service-based rather than cost-based as in the traditional VRP. Campbell et al. (2008) demonstrate empirically that the optimal solutions could be significantly different for these two objectives.

While there is a tremendous amount of work devoted to the classical VRP, the literature on the CCVRP is relatively scarce and very recent. Ngueveu et al. (2010) are among the first to solve the CCVRP. The authors proposed a metaheuristic method using two memetic algorithms (MA) and tested them on the classical VRP instances from Christofides et al. (1979) and Travelling Repairman Problem instances from Salehipour et al. (2008). Later, an adaptive large neighbourhood search (ALNS) heuristic was presented by Ribeiro and Laporte (2012) who discussed seven removal strategies and three insertion strategies to solve the instances from Christofides et al. (1979) and Golden et al. (1998). It is reported that the ALNS outperforms the MA in terms of solution quality and computational times. Ke and Feng (2013) also attempted to solve the CCVRP with a two-phase metaheuristic in which the first phase focuses on partitioning the customers by using inter-route moves whereas the second phase attempts to optimise each individual route. The first exact algorithm for the CCVRP was developed more recently by Lysgaard and Wøhlk (2014), where a branch-and-cut-and-price method was adopted. Instances from the classical VRP with up to 69 vertices are solved optimally and lower bounds are also provided for the others.

Related studies include the CCVRP with multiple trips of a single vehicle (see Martínez-Salazar

et al., 2015; Rivera et al., 2016), multiple trips with multiple vehicles (Rivera et al., 2014, 2015) and a flow (or demand) based cumulative VRP by Kara et al. (2008) and Cinar et al. (2016). In the latter two, the flow is cumulated or diminished along the tour instead of the distance being accumulated. The CCVRP can be reduced to the travelling repairmen problem (TRP) by relaxing the vehicle capacity restriction. The TRP is also known as the minimum latency problem, the travelling deliveryman problem, and the cumulative travelling salesman problem. See Salehipour et al. (2008) and Silva et al. (2012) for the single TRP; and Fakcharoenphol et al. (2007) and Luo et al. (2014) for the case of multiple TRP.

While most of the literature on the CCVRP investigate only the min-sum objective, an interesting alternative would be to explore the min-max objective, which aims to minimise the latest arrival time at the customer. The main difference between the two objectives is that the min-sum evaluates all routes whereas the min-max considers the bottleneck route with the latest arrival time only. As this problem would also be explored in this paper, a brief review on existing work is provided. To our knowledge, the only paper that investigated both min-sum and min-max objectives is Campbell et al. (2008). However in their study, the vehicle capacity and the number of vehicles are predefined with the number of customers considered being relatively small (i.e., < 100). Based on their limited empirical study, it was observed that both problems yield similar solutions for the case of one vehicle but not in the case of multiple vehicles. As a by-product, our study will also explore the impact of these two objective functions further by using a number of large VRP instances.

The min-max objective can also be applied to the school bus routing with the objective that minimises the maximum riding time of a student in a bus (Corberán et al., 2002; Pacheco and Mart, 2006; Park and Kim, 2010). Besides, López-Sánchez et al. (2014) also solved a similar real-life problem which they referred to as the balanced open VRP using the multi-start algorithm besides providing solutions to 19 school-bus routing benchmark problems from the literature. Applegate et al. (2001) also studied the classical VRP with min-max objective and proposed a branch-and-cut approach to solve a competition problem. Since the min-max objective only considers the longest route with the latest arrival time, several optimal solutions with the same objective function value could be obtained (Golden et al., 2014). This property can be made use of as the decision maker will have the added flexibility in selecting the next appropriate optimal configuration based on other non-quantifiable measures if need be.

It is worth noting that there are two definitions of the maximum arrival time in the literature (Golden et al., 2014). These include (a) the time when the last location (customer) is served (Campbell et al., 2008), and (b) the time when the vehicle arrives at the depot (Van Hentenryck et al., 2010). Definition (a) is useful in the context of delivery where there is no time pressure for the vehicle to return to the depot after serving the last customer, whereas (b) is more suitable for pick-up problems including emergency evacuation situations. In this study, we consider (a)

as it is commonly used in the existing CCVRP literature.

The contribution of this paper is fourfold:

(i) To propose an effective AVNS algorithm for the CCVRP and present new best results. We also tested our algorithm on the large instances besides the small and medium instances considered in the existing literature;

(ii) To demonstrate the flexibility and effectiveness of the AVNS algorithm to solve a related routing problem, the min-max CCVRP;

(iii) To present for the first time a large set of results for the min-max CCVRP;

(iv) To explore the managerial insights for decision making when considering these two objectives.

The rest of the paper is organized as follows. Section 2 presents the proposed AVNS algorithm for the CCVRP. Explanation of the main steps of the AVNS algorithm is provided in Section 3. This is followed by our computational results in Section 4. A related routing problem with an alternative objective, namely the min-max CCVRP is discussed in Section 5. Necessary adaptations of the algorithm and results for the min-max CCVRP are also presented in this section. In Section 6, for decision making purposes, managerial insight of considering the two objective functions is highlighted. Section 7 discusses the effects of doubling up the number of runs on both the min-sum and the min-max CCVRP results. Finally, Section 8 concludes our findings.

2. Problem statement and the proposed algorithm

The CCVRP is defined on a complete undirected graph $G = (V, E)$, where $V = \{0, 1, ..., n, n+1\}$ is a set of vertices with the first and the last vertices (0 and $n + 1$) correspond to the depot, and $V' = V \backslash \{0, n + 1\}$ is the customers set. The edge set $E(i, j) : i, j \in V, i < j$ denotes the edges connecting all the vertices. Each edge $(i, j) \in E$ is associated with a distance d_{ij}. It is assumed that the distance is equivalent to the travel time and cost, therefore they are used interchangeably throughout this paper. Each customer $i \in V'$ is associated with a non-negative demand, q_i and must be visited only once. The limited set of homogeneous vehicles is defined by R and each vehicle has capacity Q. The initialization of $|R|$ will be discussed in Section 3.1. Let t_i^k be the arrival time of vehicle k at customer i. The aim of the CCVRP is to define a set of vehicle routes that start and end at the depot where the demand of each route does not exceed the vehicle capacity. The objective function is the minimisation of the sum of arrival times at

the customers, which is given as follows:

$$\min C(x) = \sum_{k \in R} \sum_{i \in V'} t_i^k \qquad \text{subject to: } \Lambda \tag{1}$$

where Λ refers to the set of constraints which includes $t_i^k \geq 0, k \in R, i \in V'$. The mathematical formulation for this problem can be found in Ngueveu et al. (2010).

In this study, we propose an adaptive variable neighbourhood search (AVNS) with a guided diversification based on large neighbourhood search (LNS). The basic idea of VNS (Mladenović and Hansen, 1997) is to change neighbourhoods systematically while using a local search to get into the corresponding local optimum.

Our AVNS, as presented in Algorithm 1, consists of two stages where Stage 1 represents the learning stage whereas Stage 2 is the multi-level K^{th} best improvement VNS. The purpose of Stage 1 is to gather the useful information through learning to identify the importance of each of the local search operators. Stage 2, on the other hand, incorporates such information to intelligently select a smaller number of operators at each iteration when solving the problem. In Stage 1, the best improvement VNS is adopted and the frequency of success of each local search operator is recorded. Stage 2 is different from Stage 1 in two ways: (i) a smaller number of local search operator is selected pseudo-randomly based on the information recorded in Stage 1, (ii) the multi-level K^{th} best improvement approach is used as the local search engine.

Note that the K^{th} best improvement strategy is a compromise strategy that sits between the first and the best improvement strategies. The former though quicker could yield a marginal improved solution only whereas the latter may require an excessive computational time without a significant improvement. The K^{th} strategy aims to yield a good quality solution while requiring a reasonable amount of time (see Salhi, 2017 for more details). In our K^{th} best improvement strategy, K improving moves are evaluated and the best one selected.

While most of the AVNS in the literature focus on adapting the shaking step in the VNS (see Stenger et al., 2013, Li et al., 2015, Hof et al., 2017) or in the diversification stage in Wei et al. (2014) to select one of the few diversification mechanisms probabilistically, our algorithm applies an intelligent selection mechanism in the local search step instead. Very recently, the AVNS has shown to be successful in producing very promising results for the classical VRP (Sze et al., 2016). We aim to adopt this effective adaptive search algorithm by introducing some new interesting features that relate to the CCVRP.

2.1. Stage 1 of the AVNS algorithm

At the beginning of Stage 1 (line 1 in Algorithm 1), we first define a set of neighbourhood structures S_p, for $p = 1, ..., p_{max}$, local search operators, L_h, for $h = 1, ..., h_{max}$, and set the score

Algorithm 1 The AVNS algorithm (Stage 1).

Require: Initial solution x.

1: Define $MaxDiv$, S_p; $p = 1, ..., p_{max}$, and L_h; $h = 1, ..., h_{max}$. Initialise $Score(L_h) = 0$, $\lambda = \lambda_{min}$, $numDiv = 1$ and $x_{best} = x$.
2: **while** $(numDiv \leq MaxDiv)$ **do**
3: Set $p \leftarrow 1$.
4: **while** $(p \leq p_{max})$ **do**
5: *Shaking*: Generate x' from the p^{th} neighbourhood of x (i.e., $x' \in S_p(x)$).
6: *Best improvement local search*: For each L_h, find $Gain(L_h)$.
7: Select the maximum gain, $\max_{1 \leq g \leq h_{max}} Gain(L_g)$ and perform the move to obtain x''.
8: *Move or not*:
9: **if** $(C(x'') < C(x))$ **then**
10: $x \leftarrow x''$ and $p \leftarrow 1$.
11: **else**
12: $p \leftarrow p + 1$.
13: **end if**
14: **end while**
15: **if** $(C(x) < C(x_{best}))$ **then**
16: $x_{best} \leftarrow x$ and $\lambda \leftarrow \lambda_{min}$.
17: **else**
18: $\lambda \leftarrow \lambda + 0.05n$.
19: **end if**
20: *Diversification*: Apply the LNS based on λ to get $\tilde{x}$.
21: **if** $(C(\tilde{x}) < C(x_{best}))$ **then**
22: $x_{best} \leftarrow \tilde{x}$.
23: **end if**
24: Set $numDiv \leftarrow numDiv + 1$ and $x \leftarrow \tilde{x}$.
25: **end while**
26: *Learning*: For each operator L_h, compute the probability, $Prob(L_h)$ and cumulative probability, $F(L_h)$ for $h = 1, ..., h_{max}$.

(*Cont.*)

of the operator, $Score(L_h) = 0$. An initial solution, x is generated and used as the global best, x_{best}. While the stopping criterion is not met, the main steps of the VNS are repeated iteratively (lines 2–26). In the shaking step (line 5), a random solution x' is generated from $S_1(x)$ and then improved by the best improvement procedures iteratively to obtain x''. Here, for each operator L_h, we find the $Gain(L_h)$ in performing the move where more details are provided in Section 3.5.3. The operator with the maximum gain is selected and performed to improve the solution x' (i.e., $L_h = \text{Arg} \max_{1 \leq g \leq h_{max}} Gain(L_g)$). In addition, the score of operator L_h is computed by adding the current score to the marginal relative gain as shown in Equation (2).

$$Score(L_h) = Score(L_h) + \frac{Gain(L_h)}{\max_{1 \leq g \leq h_{max}} Gain(L_g)}, \qquad h = 1, ..., h_{max} \tag{2}$$

Algorithm 1 The AVNS algorithm (Stage 2).

27: Define l_{cmin}, l_{cmax}, $MaxDiv2$ and initialise $\lambda = \lambda_{min}$, $nonImproveDiv = 0$.
28: **while** ($nonImproveDiv \leq MaxDiv2$) **do**
29: Set $p \leftarrow 1$.
30: **while** ($p \leq p_{max}$) **do**
31: *Operator selection from learning*: Randomly generate $m_{max} \in [l_{cmin}, l_{cmax}]$.
32: **for** ($m = 1$ to m_{max}) **do**
33: Generate $\alpha \in (0, 1)$, find $\hat{L}_h = F^{-1}(\alpha)$ with F defined in Equation (3)
34: Choose the operator based on $\hat{L}_h$ and set $W_m \leftarrow \hat{L}_h$.
35: **end for**
36: Sort the operator W_m; $m = 1, ..., m_{max}$ based on their complexity.
37: *Shaking*: Generate x' from the p^{th} neighbourhood of x (i.e., $x' \in S_p(x)$).
38: *Multi-level K^{th} best improvement local search*:
39: Set $m \leftarrow 1$.
40: **while** ($m \leq m_{max}$) **do**
41: Find $Gain(W_m)$ using the K^{th} best improvement and perform the move to obtain x''.
42: **if** ($C(x'') < C(x')$) **then**
43: $x' \leftarrow x''$ and $m \leftarrow 1$.
44: **else**
45: $m \leftarrow m + 1$.
46: **end if**
47: **end while**
48: *Move or not*:
49: Same as lines 9–13 except x' is used instead of x''.
50: **end while**
51: **if** ($C(x) < C(x_{best})$) **then**
52: $x_{best} \leftarrow x$, $\lambda \leftarrow \lambda_{min}$ and $nonImproveDiv \leftarrow 0$.
53: **else**
54: $\lambda \leftarrow \lambda + 0.05n$ and $nonImproveDiv \leftarrow nonImproveDiv + 1$.
55: **end if**
56: *Diversification*: Apply the LNS based on λ to get $\tilde{x}$.
57: Same as lines 21–23 and set $x \leftarrow \tilde{x}$.
58: **end while**

If the local minimum x'' is better than the incumbent best solution x, we set $x = x''$ and the search reverts back to the first neighbourhood S_1, otherwise the search will explore the next neighbourhood (i.e., $p = p+1$) where the VNS steps are then repeated (lines 9–13). This process continues until p reaches p_{max} (lines 4–14).

After all the neighbourhoods are explored, the incumbent x is compared against the global best, x_{best} (lines 15–19). If x is better than x_{best}, we set $x_{best} = x$ and the diversification control parameter, $\lambda = \lambda_{min}$; otherwise λ is increased by $0.05n$. This parameter is used to guide the large neighbourhood search (LNS) diversification. A new perturbed solution is then created using the LNS (line 20) and the process reverts back to S_1 to repeat the VNS. This LNS-based strategy will be revisited in Section 3.8.

The search stops when a predefined maximum number of diversifications is reached. Stage 1 is ended by the learning step (line 27) where the probability and cumulative probability for each operator are computed using Equation (3). Such information is used in Stage 2 for the selection of the operators.

$$Prob(L_h) = \frac{Score(L_h)}{\sum_{h=1}^{h_{max}} Score(L_h)} \qquad \text{and} \qquad F(L_h) = \sum_{L_h \leq L_{h_{max}}} Prob(L_h). \tag{3}$$

2.2. Stage 2 of the AVNS algorithm

In Stage 2, the parameters such as λ, the number of operators to be selected and the stopping criterion are first initialized. We use a fixed number of non-improved diversifications (i.e., $MaxDiv2$) as our termination criterion. While such criterion is not met, the adaptive VNS is carried out by first selecting a set of m_{max} operators pseudo-randomly based on $\hat{L}_h = F^{-1}(\alpha)$ where $\alpha \in (0, 1)$ (lines 33–36). Here, the higher the score for an operator is, the greater the chance for such an operator to be selected. Such a mechanism as shown in Figure 1, is also referred to as the inverse method in the literature (Elshaikh et al., 2015; Sze et al., 2016).

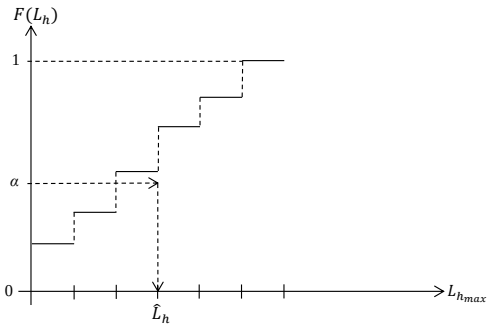

Figure 1: Selection of the operator using the cumulative probability.

The operators, W_m $(m = 1, ..., m_{max})$, are then sorted in levels such that the operator in Level 1 is the simplest, followed by Level 2 which is the second simplest until Level m_{max} the most complex one, with the order stated in Section 3.7. Next, the shaking step with the same neighbourhood structures defined in Stage 1 is performed (line 38), and followed by the multi-level K^{th} best improvement local search (lines 39–48). This local search engine is based on the multi-level metaheuristic originally presented by Salhi and Sari (1997) for the multi-depot VRP. Initially, we set $m = 1$ and apply the first level of refinement with the local search W_1 where the calculation of $Gain(W_m)$ (line 42) will be revisited in Section 3.5.3. If the solution x'', which is found based on the K^{th} best improvement selection strategy is better than x', we set $x' = x''$ and the search reverts back to $m = 1$, otherwise we proceed to the next level by setting $m = m + 1$. Since the operators are sorted based on the complexity of the move, it is worth reverting back to Level 1 where the search is relatively less computationally expensive. When the solution could not be

improved further in the current level, the next level is used with the aim to improve upon the solution. This process continues until all levels are explored (i.e., $m = m_{max}$) where the LNS diversification is performed (lines 57–58). The search terminates when there is no improvement on the solution after a maximum number of consecutive diversifications.

3. Explanation of the main steps

3.1. Initialisation of $|R|$

Ngueveu et al. (2010) pointed out that when there is an unlimited number of vehicles available, the number of routes in the optimal solution of the CCVRP is simply the total number of customers, n. Therefore, to enable a consistent comparison of our algorithm with the others, the number of vehicles is fixed as suggested in the literature. For those instances that have not been solved for the CCVRP before, we initially define the number of routes using the lower bound $|R| = \lceil \sum_{i \in V'} \tilde{q}_i / Q \rceil$ and generate the initial solution. The AVNS algorithm is then executed. If the solution obtained is infeasible after 5 runs, we add a new route to the solution and re-run the AVNS. This process is repeated until the solution is feasible or the number of routes is equal the one found in our VRP solution in Sze et al. (2016). We note that the number of routes for the CCVRP can be smaller or equal the one found in the VRP as the route length constraint is not imposed for the CCVRP.

3.2. Initial solution

A parallel greedy insertion method is adopted to generate an initial solution. In other words, we first assign the nearest $|R|$ customers to the depot as the first customer in the routes, followed by the computation of the insertion costs for the remaining customers. The customer with the lowest insertion cost (among all routes) is then assigned accordingly. In the case of a tie, the customer with a precedent index is chosen. These steps are repeated until all customers are inserted.

If a feasible insertion is not found for a customer 'a', we employ a simple perturbation strategy by considering removing an existing customer, say customer 'b' one-by-one systematically from the routes and inserting customer 'a' in this route, where this move must be a feasible move irrespective of the cost of insertion. Customer 'b' will then be inserted to another route by repeating the same process until both customers 'a' and 'b' are inserted or the termination criterion is reached. Here, we maintain a simple tabu list of size $ts = \max(5, 0.01n)$ to prevent the same customers to be deleted each time. When a customer ('a' or 'b') is inserted, it is flagged tabu so that it will not be deleted in the next ts iterations. This perturbation strategy terminates when there is no other customer that can be deleted from the routes to enable a feasible insertion of customer 'b'. For this case, customer 'b' is added to the route with the

smallest capacity violation and the route is penalized accordingly as will be discussed in Section 3.4.

3.3. Neighbourhood reduction scheme

To increase the efficiency of our AVNS algorithm in terms of computational time, we propose a neighbourhood reduction scheme which is used throughout both stages of our algorithm. For each customer i, we identify a set of neighbouring customers which could be potentially placed next to i in a route hence restricting the search to those customers only (Salhi and Sari, 1997; Sze et al., 2016). A binary matrix, $Flag \in \mathbb{R}^{(n+1)\times(n+1)}$ is defined where we set $Flag(i,j) = 1$ if customers i and j are potential customers to be placed next to each other. To define the values for $Flag$, a simple q^{th} quantile method is adopted. Here, for $i = 0, ..., n$, the top q^{th} quantile nearest customers (based on distance) to i are assigned a value 1. This mechanism has an added advantage that each vertex has approximately the same number of neighbours. For the CCVRP problem, the arrival time at the last vertex (depot) is not considered in the cost and therefore we set $Flag(i,0) = 0$ for all $i \in V$. It is worth noting that, contrarily to the VRP, this is a special asymmetric case where $Flag(0,i)$ is not necessarily equal to $Flag(i,0)$.

3.4. Penalized objective function

During the search, the algorithm only considers moves that can improve the current solution. To increase flexibility in exploring the search space, the improvement moves are extended to infeasible moves by attaching penalties in the objective function. To guide the search from diverging in case of infeasible moves, we restrict these to be within a certain violation threshold value.

Suppose C_k is the sum of arrival times for customers in route k, $\tilde{Q}_k$ the route demand and η_k the penalized cost for capacity where $k = 1, ..., |R|$. Two user-defined parameters, β and γ are introduced to control the penalized value, ζ. Equation (4) states that the maximum percentage of the capacity violation allowed in a route is restricted to remain within β, whereas γ in Equation (5) controls the percentage of increase in the total arrival times for a violated route. The value of γ must be carefully selected such that it is effective enough to penalize even the small violation. The penalized value ζ is defined in Equation (6), which is used to calculate the penalized cost, η_k in Equation (7). Finally, the augmented objective function for route k is then defined by

Equation (8).

$$\tilde{Q}_k \leq (1+\beta)Q \tag{4}$$

$$C_k + \eta_k \leq (1+\gamma)\bar{C}, \qquad \text{where } \bar{C} = \sum_{k \in R} \frac{C_k}{|R|} \tag{5}$$

$$\zeta = \gamma/\beta \times \bar{C} \tag{6}$$

$$\eta_k = max(0, \frac{\tilde{Q}_k - Q}{Q} \times \zeta) \tag{7}$$

$$C'_k = C_k + \eta_k \tag{8}$$

3.5. Evaluation of the cost of a move

For the CCVRP, to evaluate the cost of traversing from point i to j, we need to know not only the distance between i and j but also the remaining customers to be visited in the route. This is because the time spent travelling between i and j adds to the arrival times at the remaining customers in the route. Due to this special cost structure, we find it appropriate to provide the necessary route cost calculations in the following subsections.

Let $R_k^i, ..., R_k^{n_k}$ with R_k^j being the j^{th} customer in route k, $j = 1, ..., n_k$, $k = 1, ..., |R|$. The total cost of route k is given as the sum of the n_k arrival times of the customers along the route:

$$C_k = \sum_{i=R_k^1}^{R_k^{n_k}} t_i^k, \qquad k = 1, ..., |R| \tag{9}$$

Since it is computationally expensive to calculate all the relevant values for each move, some information is pre-computed and is updated as the search progresses. For each route k, we compute the cumulative cost, t_i^k up to the i^{th} customer in the route.

3.5.1. Cost variation of removing customer(s)

The cost variation when u customers are removed, starting from the i^{th} customer until the $(i+u-1)^{th}$ in route k is given as:

$$\psi_k^-(u) = \sum_{p=R_k^i}^{R_k^{i+u-1}} t_p^k + (n_k - (i+u-1)) \times \Delta G^+(k,u,i), \tag{10}$$

$$\text{where } \Delta G^+(k,u,i) = \sum_{a=R_k^{i-1}}^{R_k^{i+u-1}} d_{a,a+1} - d_{R_k^{i-1}, R_k^{i+u}}$$

The first part of Equation (10) is the sum of the cumulative costs for the u customers that are to

be removed, which are already computed beforehand. The second part of the equation is based on the original gain of this move, $\Delta G^+(k,u,i)$ (i.e., as the one in the classical VRP) and the remaining customers in the route. The calculation of the route cost using this cost variation is explained in Section 3.5.3.

To speed up the algorithm, the cost variation of removing one ($u = 1$) and two customers ($u = 2$) are computed beforehand by Equation (10) and stored using a special data structure. This is motivated by the idea of using preprocessed information for the evaluation of moves proposed by Vidal et al. (2014) which has also been successfully implemented in Sze et al. (2016). To increase the efficiency of the data structure proposed in Sze et al. (2016), another memory structure that stores the cost variation of removing two customers, in addition to the removal costs for one customer only is also added. In other words, the cost variation of removing each customer, $Z_1 \in \mathbb{R}^n$ and the cost variation of removing two consecutive customers, $Z_2 \in \mathbb{R}^n$ from the routes are stored. Z_1 is used for all the operators when a single customer is removed from a route (i.e., the 1-insertion and the 1-1 exchange), whereas Z_2 is applied for the operators that involve the removal of two customers (i.e., the 2-insertion and the 2-1 interchange), see Section 3.6 and Section 3.7 for details. When a customer in route k has changed its position, the values of Z_1 and Z_2 for all customers of route k need to be updated. The use of this data structure speeds up the evaluation of the moves process as this information can be used directly when needed without the re-computation of such information again.

3.5.2. Cost variation of inserting customer(s)

The cost variation of inserting u customers (which is removed from route k) at position j in route l is given in Equation (11), where t'^l_p is defined as the new cumulative distances of the u customers to be inserted in route l.

$$\psi_l^+(u) = \sum_{p=R_k^i}^{R_k^{i+u-1}} t'^l_p - (n_l - j + 1) \times \Delta G^+(k,u,i,l,j), \tag{11}$$

$$\text{where } \Delta G^+(k,u,i,l,j) = d_{R_l^{j-1},R_l^j} - (d_{R_l^{j-1},R_k^i} + \sum_{a=R_k^i}^{R_k^{i+u-2}} d_{a,a+1} + d_{R_k^{i+u-1},R_l^j})$$

Note that in Equation (11), contrary to Equation (10), the remaining customers do not change regardless of the number of customers inserted.

3.5.3. Evaluating the route cost

When evaluating the cost of a move, the change of the penalty cost needs also to be computed as shown in Equation (12).

$$\Delta\eta_k = \eta_k^{old} - \eta_k^{new}, \qquad k = 1, ..., |R| \tag{12}$$

For a particular move in any route k, the gain of the route cost, $\Delta C_k(u, w)$ is computed by removing 'u' of its customers and receiving 'w' customers from another route (route l) as given in Equation (13). This is performed in a similar way for route l where w customers are removed while u customers are inserted.

$$\begin{aligned} \Delta C_k(u, w) &= \psi_k^-(u) - \psi_k^+(w) + \Delta\eta_k, \\ \Delta C_l(w, u) &= \psi_l^-(w) - \psi_l^+(u) + \Delta\eta_l \end{aligned} \tag{13}$$

In Algorithm 1, the operators L_h in Stage 1 (line 6) and W_m in Stage 2 (line 42) are represented by u and w (e.g., 1-insertion means $u = 1$ and $w = 0$ while 2-1 exchange refers to $u = 2$ and $w = 1$ for route k and vice versa for route l). For a move that involves two routes (k and l), $Gain(L_h)$ is the maximum gain obtained by operator L_h which is defined by a move that removes u customers from route k to be inserted to route l and receives w customers from route l to be inserted into route k as shown in Equation (14). Likewise, the K^{th} maximum gain in Stage 2, $Gain(W_m)$ is represented in Equation (15), where Kmax denotes the maximum value among the K gains.

$$Gain(L_h) = \max_{k,l \in R}\{\Delta C_k(u, w) + \Delta C_l(w, u)\} \tag{14}$$

$$Gain(W_m) = \mathrm{Kmax}_{k,l \in R}\{\Delta C_k(u, w) + \Delta C_l(w, u)\} \tag{15}$$

Here, a positive value of $Gain(L_h)$ or $Gain(W_m)$ indicates an improvement move whereas a negative value displays a non-improving move. Note that Equation (13) is used in the evaluation of moves in both the shaking and the local search steps in Stage 1 and Stage 2 of the AVNS algorithm (Algorithm 1).

Finally, the new route cost for route $\bar{k}$ which is affected by a move due to operators L_h or W_m is obtained by Equation (16).

$$C_{\bar{k}} = C_{\bar{k}} - \Delta C_{\bar{k}}, \tag{16}$$

3.5.4. Effect of the reversed route

One important characteristic of the CCVRP is that the route cost depends not only on the edges, but also on the order of the customers in the solution. In other words, the route cost for any route may be different when the route is reversed. Besides, a reduced cost is obtained when the

shorter edges are visited first followed by the longer ones as illustrated in Figure 2. It is therefore necessary to compute the cost of the reversed route to check whether or not the route is worth reversing.

Let $\tilde{C}_k$ be the cost of route k in reversed order and $D_k = \sum_{j=R_k^1}^{R_k^{n_k+1}} d_{j-1,j}$ the route length in the classical VRP. $\tilde{C}_k$ can then be expressed in Equation (17) as given by Ngueveu et al. (2010):

$$\tilde{C}_k = n_k D_k - C_k \tag{17}$$

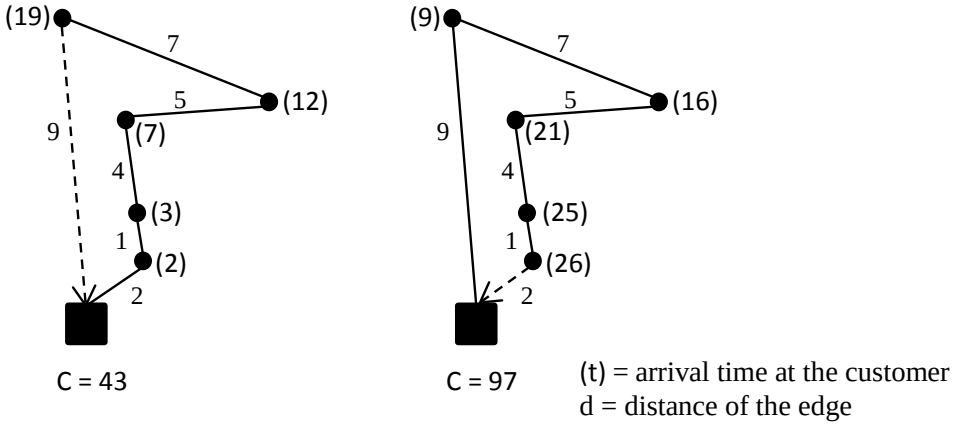

Figure 2: An example of a reversed route in the CCVRP.

3.6. Neighbourhood structures

In this study, five neighbourhood structures are defined in the following order: S_1: 2-insertion; S_2: 2-1 interchange; S_3: segment reshuffle; S_4: cross-exchange and S_5: head swap. All neighbourhoods except for S_3 are inter-route moves. Here, the moves are conducted by randomly choosing a route (donor) and its customer(s) and then moving them to other route(s) (receiver) by considering the positions in the receiver route(s) systematically until the first feasible move is obtained.

2-insertion: In this neighbourhood, two consecutive customers are removed from a randomly selected route. The customers are then split and inserted into two different receiver routes separately.

2-1 interchange: The 2-1 exchange operator first exchanges one of the two customers removed from a randomly selected route with a customer from the first receiver route and then inserts the second customer into the other receiver route.

Segment reshuffle: Since the order of the customers in a route has a significant effect on the cost, this neighbourhood aims to shake the customers order. For this move, a segment of

$\tilde{s}_1$ customers in a randomly selected route (k) is reshuffled, where $\tilde{s}_1$ is randomly chosen in the range of $[0.5n_k, 0.7n_k]$.

Cross-exchange: This neighbourhood attempts to exchange a segment of $\tilde{m}_1$ customers in the donor route (j) with a segment of $\tilde{m}_2$ customers from the receiver route (k). The values of $\tilde{m}_1$ and $\tilde{m}_2$ are randomly chosen in the range of $[0.2n_j, 0.4n_j]$ and $[0.2n_k, 0.4n_k]$ respectively.

Head swap: This neighbourhood intends to shake the first half of the routes since a good solution of the CCVRP tends to have shorter edges in the early part of the route. Here, the first half segment (head) of a route is swapped with the head from the other route where the size of the heads may be different.

3.7. Local search engine

The local search engine consists of six operators: (i) 1-insertion, (ii) 1-1 exchange, (iii) 2-insertion, (iv) 2-opt, (v) 2-opt* and (vi) cross-tail. The first three operators consider both intra and inter-routes moves, the 2-opt is an intra-route operator whereas the last two consider only inter-routes moves. In Stage 1 of the AVNS algorithm, all six operators are carried out and the move with the overall best improvement is performed in each iteration. On the other hand, a subset of these operators is selected in Stage 2 with the best value among K improvements is selected.

1-insertion: In this operator, a customer is removed from its position and reinserted elsewhere either in the same or in a different route.

1-1 exchange: This operator aims to swap two customers from two different positions or routes in which the positions of removal and insertion are not necessarily the same.

2-insertion: Here, we remove two consecutive customers from a route and insert them elsewhere following the original or the inverted sequence.

2-opt: The 2-opt aims to remove a pair of non-adjacent edges and replace with a new pair in the same route (Lin, 1965).

2-opt*: This operator considers deleting and replacing two edges in two different routes.

Cross-tail: In this operator, the tail of a route of all possible sizes are considered. This tail is then exchanged with the tail from another route where the sizes of the two tails are not necessarily the same and the order of the customers may be reversed (Sze et al., 2016).

3.8. A LNS diversification strategy with a VNS structure

In both stages of the AVNS algorithm, a large neighbourhood search (LNS) diversification strategy is adopted when all the neighbourhoods are being explored. Such a strategy, which can been seen as a more powerful shaking procedure, aims to search in other promising regions

that may not have been visited before. Using the LNS, some customers are removed and then reinserted back to the routes to produce a new solution. As the success of this algorithm depends highly on the intensification of the removal strategy, the perturbation intensity has to be carefully monitored so that the algorithm is effective in guiding the search toward escaping from the local optimum while not degrading the solution to a random restart point. Here, the LNS which we propose follows a VNS structure. The diversification control parameter, λ, represents the number of customers to be deleted from the routes which is defined in the range of $[\lambda_{min}, \lambda_{max}] = [max(10, 0.1n), min(300, 0.3n)]$. Initially, we set $\lambda = \lambda_{min}$ and its value is gradually increased if no better solution is found. Once a new best solution is obtained, λ is reinitialized to the minimum value. We consider four removal and two insertion strategies. In the diversification step, one from each of the removal and the insertion strategies are chosen at random.

Random removal

In this removal strategy, a set of λ customers are randomly selected and removed from the routes. The random removal tends to result in a poorer solution but it gives a reasonable diversification to the solution.

Worst removal

Suppose $C(x)$ is the cost of the solution x and $C_{-i}(x)$ is the cost after customer i is deleted, the gain(i) is defined as the difference between the cost when customer i is in, and removed from, the solution. This is given as gain(i) $= C(x) - C_{-i}(x)$. The customer is chosen based on its corresponding gain ratio which is stated as follows:

$$\text{ratio}(i) = \text{demand}(i)/\text{gain}(i)$$

To avoid a deterministic removal, the customers are selected based on probability values (Ropke and Pisinger, 2006). Let $\tilde{L}$ be the array of all customers sorted in descending ratio, z be a uniformly distributed random number chosen in the range $[0, 1)$, and θ is a user-defined parameter. The algorithm then selects customer $a = \tilde{L}[\lceil z^{\theta} n \rceil]$ from array $\tilde{L}$ and removes it. This process is repeated until λ customers are deleted from the solution. Here, the value of θ is defined such that the customer with a small demand and a high gain has a higher chance to be selected. Since array $\tilde{L}$ is sorted in descending order, we set $\theta < 1$ leading to z^{θ} closer to 1 which will guarantee a high value of $z^{\theta} n$ (i.e., closer to n). A similar sorting scheme can also be used by sorting the ratio in ascending order while setting $\theta > 1$. As an illustration example, we set $\theta = 0.2$ (this is the value we used in our experiments) and suppose $n = 100$. When $z = 0.3$, $\tilde{L}[79]$ will be selected and for $z = 0.8$, $\tilde{L}[96]$ is deleted. This rule can also be considered as a simple look-ahead strategy to allow for more freedom at the insertion stage.

Worst-distance removal

The worst-distance removal simply computes the distances between every edge in the routes and sorts them in increasing order. The selection of customers that are connected by the edges is performed at random, using the probability method as previously described.

Conflicting sector removal

This removal strategy starts by dividing the route plane into different sectors based on the angle from the depot where the vertices are given in X-Y coordinates, and computing the number of routes contained in each sector. The sector which has the largest number of routes is identified and λ customers in the sector are then removed. When there are two or more sectors having an equal number of routes, the tie is broken by choosing one of these sectors randomly. To provide diversity when defining the sectors, the initial point of computing the angle from the depot starts at a randomly chosen customer which can be considered in clockwise or anti-clockwise direction.

Basic greedy insertion

In the repair phase, we apply the basic greedy least-cost insertion. Let $\Delta f_{i,k}$ be the change in the objective value when customer i is inserted into route k at the position that increases the objective function the least. We set $\Delta f_{i,k} = \infty$ if customer i cannot be inserted into route k. Let Ω be the set of customers that is removed using the LNS removal strategy. For each unassigned customer $i \in \Omega$, the algorithm computes $\Delta f_{i,k}$ and then inserts customer i that yields $\min_{i\in\Omega}\Delta f_{i,k}$. This process is repeated until all customers in Ω are assigned to the routes. For the case when a feasible insertion is not found, the same perturbation strategy, as discussed in Section 3.2, is applied.

Regret cost insertion

The regret cost insertion tries to improve the myopic behaviour of the greedy insertion (Ropke and Pisinger, 2006). Here, for each removed customer i, the regret value is defined as the difference between the cost of inserting customer i to the best position and the second best position. The customer with the highest regret value is chosen to be inserted in each iteration, until all the customers are added back to the routes. This rule is commonly used in the generation of an initial solution for the classical transportation problem (i.e., Vogel Approximation Method).

4. Computational experiments for the CCVRP

The proposed AVNS algorithm is coded in C++ and run on a Intel Core i7 3.4 GHz PC with 8 GB RAM. The effects of diversification and Stage 2 in the AVNS algorithm are first investigated, followed by a full experiment to evaluate the performance of our algorithm compared to the

ones published in the literature. Four VRP benchmark data sets which consist of the small (Christofides et al., 1979; Fisher, 1994), medium (Golden et al., 1998) and large (Li et al., 2005) instances are used for computational experiments. They are referred to as Set 1a, Set 1b, Set 2 and Set 3 hereafter. The data points, which are given in X-Y coordinates in Set 1a and Set 1b are randomly distributed whereas instances in Set 2 and Set 3 are based on some geometry structures (concentric circles, squares and stars). There are 7 and 20 instances in Set 1a and Set 2 respectively, where the results are reported in the CCVRP literature. However for Set 1b and Set 3 which consist of 3 smaller and 12 larger instances respectively, there are no results available for comparison as these are tested for the CCVRP for the first time. For convenience, all the data can be downloaded from CLHO (2016).

Preliminary experiments show that $q^{th} = 3\%$ is sufficient to be used in the neighbourhood reduction scheme. In Stage 2 of the AVNS algorithm, we define $K = 3$ for the K^{th} best improvement and the range for the number of local search operators, $[l_{cmin}, l_{cmax}] = [3, 5]$. For the penalized objective function, $\beta = 0.05$ and $\gamma = 0.60$ are used. We set the parameter, θ in the worst removal and worst-distance removal of the LNS to be 0.2, whereas the angle used to define the sector in the conflicting sector removal is $\pi/12$.

For comparisons purpose, we present, as previously shown in the literature, the best and the average results obtained by our algorithm together with the results found in the CCVRP papers. Moreover, the relative percentage deviation: $((C_{Best} - C_{BKS})/C_{BKS}) \times 100$ is computed for each instance, where C_{Best} and C_{BKS} represent the best result found using our algorithm and the best known solution value reported in the literature respectively.

4.1. The effects of diversification and Stage 2 in the AVNS algorithm

To study the effect of diversification, we first run the algorithm by fixing the stopping criterion as four diversifications for Stage 1 and four non-improved diversifications for Stage 2 and record the time consumed. The same amount of time is used to execute the algorithm without diversification. Similarly, we run Stage 1 only by fixing the running time at the same value as the one consumed when the algorithm is executed fully for both stages. This test is performed over 5 runs for each variant in Set 1–Set 3 and the results are presented in Table 1 with $C^{(1)}_{Avg}$, $C^{(2)}_{Avg}$, C_{Avg}, T_{Avg} refer to the average cost without diversification, without Stage 2, average cost and time with both diversification and Stage 2 respectively. It is apparent that the use of both of these strategies achieves better solution compared to the one without. More specifically, the results are improved by 2.72% when the diversification is included and 1.64% when Stage 2 is executed in the algorithm. In addition, the effects are more significant for medium and large-sized test instances. This experiment illustrates the effectiveness of the diversification strategy and the use of Stage 2 making such additions worthwhile to be included.

Table 1: The effects of diversification and Stage 2 in the AVNS algorithm.

	No Diversification	No Stage 2	With Diversification & Stage 2	
	$C^{(1)}_{Avg}$	$C^{(2)}_{Avg}$	C_{Avg}	T_{Avg} (s)
Set 1a	4431.54	4357.60	4348.91	42.97
Set 1b	3637.45	3594.70	3562.64	8.99
Set 2	66360.17	65776.93	64804.86	570.62
Set 3	921002.35	912138.03	886431.52	2190.55
Gap (%)	2.72	1.64		

Gap (%) $= (C^{(p)}_{Avg} - C_{Avg})/C_{Avg} \times 100$, for $p = 1, 2$.

4.2. Comparisons with existing CCVRP algorithms

In this section, we present the results obtained using the AVNS algorithm compared to the existing three algorithms available in the literature. The stopping criterion we apply is four diversifications for Stage 1 and four non-improved (consecutive) diversifications for Stage 2. This setting is found after some preliminary experiments.

The computational results for Set 1a and Set 2 are summarized in Table 2 and Table 3 respectively. Our results are shown in the last three columns under AVNS. For each instance, we show the best and the average solutions obtained by our algorithm over 5 runs, such that they are consistent with the other algorithms in the literature that use the same number of runs. It is worth mentioning that the C_{Best} reported is the best result obtained in 5 independent runs with each run using the same initial solution. The deviation of the best solution from the BKS value is stated as Dev. The gap between the average solution and the best solution obtained ($\text{Gap}_{robustness}$) is also reported here. For instances where new solutions are found, the values of these new results are used as the BKS instead.

For Set 1a, our solutions reach all previously best known solutions while identifying one new best result (i.e., C5), with a 0.06% improvement over the previous best solution obtained by the two-phase metaheuristic (Ke and Feng, 2013). Optimal solutions were reported for five instances, namely C1–C4 and C12, by Lysgaard and Wøhlk (2014). These instances are marked with a '*' in Table 2, where our results match all these optimal values. Besides, based on the lower bounds (LB) provided in the literature (see Lysgaard and Wøhlk, 2014), the optimality gap of our solutions for C5 and C11 are 0.53% (LB=5775.28) and 1.62% (LB=7197.69) respectively. These results demonstrate that the AVNS is very effective at solving the CCVRP. In addition, the gap between the average solution and the best solution is just about 0.35%, which indicates that our algorithm is very robust for tackling this class of routing problems.

Table 2: Results for Set 1a (Christofides et al., 1979) instances.

			MA1			MA2			ALNS			2-phase meta			AVNS		
#(n)	|R|	BKS	C_{Best}	C_{Avg}	Dev (%)	C_{Best}	C_{Avg}	Dev (%)	C_{Best}	C_{Avg}	Dev (%)	C_{Best}	C_{Avg}	Dev (%)	C_{Best}	C_{Avg}	Dev (%)
C1(50)	5	*2230.35	**2230.35**	**2230.35**	0.00	**2230.35**	2245.74	0.00	**2230.35**	2235.27	0.00	**2230.35**	2240.21	0.00	**2230.35**	2243.82	0.00
C2(75)	10	*2391.63	2421.90	2443.07	1.27	2429.18	2556.24	1.57	**2391.63**	2401.72	0.00	**2391.63**	2430.48	0.00	**2391.63**	2393.16	0.00
C3(100)	8	*4045.42	4073.12	4073.12	0.68	4073.12	4079.69	0.68	**4045.42**	4063.98	0.00	**4045.42**	4064.22	0.00	**4045.42**	4075.25	0.00
C4(150)	12	*4987.52	**4987.52**	5020.75	0.00	**4987.52**	4996.84	0.00	**4987.52**	4994.93	0.00	**4987.52**	5001.67	0.00	**4987.52**	4993.57	0.00
C5(199)	17	5806.02	5810.12	5842.00	0.07	5810.12	5840.77	0.07	5838.32	5857.76	0.56	5809.59	5856.46	0.06	**5806.02**	5836.75	0.00
C11(120)	7	7314.55	7317.98	7395.83	0.05	7347.49	7395.46	0.45	7315.87	7341.28	0.02	**7314.55**	7334.09	0.00	**7314.55**	7335.51	0.00
C12(100)	10	*3558.92	**3558.92**	3559.23	0.00	3559.43	3559.43	0.01	**3558.92**	3566.06	0.00	**3558.92**	3561.11	0.00	**3558.92**	3567.90	0.00
Average			4342.84	4366.34	0.30	4348.17	4382.02	0.40	4338.29	4351.57	0.08	4334.00	4356.45	0.01	**4333.49**	4348.91	**0.00**
#Best			3			2			5			6			**7**		
#Worst			1			4			1			0			0		
$Gap_{robustness}$ (%)			0.45			1.06			0.30			0.61			0.35		

*optimal solution (Lysgaard and Wøhlk, 2014),
MA1 and MA2: Ngueveu et al. (2010), ALNS: Ribeiro and Laporte (2012), 2-phase meta: Ke and Feng (2013), Dev = $(C_{Best} - C_{BKS})/C_{BKS} \times 100$, $Gap_{robustness} = (C_{Avg} - C_{Best})/C_{Best} \times 100$

Table 3: Results for Set 2 (Golden et al., 1998) instances

			MA1			MA2			ALNS			2-phase meta			AVNS		
#(n)	|R|	BKS	C_{Best}	C_{Avg}	Dev (%)	C_{Best}	C_{Avg}	Dev (%)	C_{Best}	C_{Avg}	Dev (%)	C_{Best}	C_{Avg}	Dev (%)	C_{Best}	C_{Avg}	Dev (%)
G1(240)	9	54683.36	54815.17	54878.25	0.24	54826.53	54917.44	0.26	54786.92	54853.76	0.19	**54683.36**	54752.56	0.00	54749.30	54789.30	0.12
G2(320)	10	100560.00	100836.90	100918.54	0.26	100800.33	100924.25	0.22	100662.53	100934.34	0.08	100666.06	100780.92	0.09	**100560.00**	100638.10	0.00
G3(400)	10	170924.00	171277.26	171400.35	0.21	171311.81	171523.42	0.23	171613.59	172231.14	0.40	171040.04	171330.48	0.07	**170924.00**	171098.60	0.00
G4(480)	10	261993.00	262584.23	262830.96	0.23	262646.37	263020.99	0.25	263433.03	265207.46	0.55	262145.12	262587.54	0.06	**261993.00**	262178.40	0.00
G5(200)	5	114163.64	**114163.64**	114237.00	0.00	**114163.64**	114338.39	0.00	114494.66	114846.27	0.29	**114163.64**	114171.15	0.00	**114163.64**	114218.70	0.00
G6(280)	7	140430.09	**140430.09**	140456.96	0.00	140463.67	140556.57	0.02	140804.64	140929.10	0.27	**140430.09**	140530.01	0.00	**140430.09**	140547.20	0.00
G7(360)	8	179388.00	183282.64	186702.15	2.17	190371.53	192578.18	6.12	180481.56	181610.82	0.61	183509.62	184277.02	2.30	**179388.00**	180131.60	0.00
G8(440)	10	193698.00	194312.60	194510.99	0.32	194273.58	194454.29	0.30	194988.74	195174.85	0.67	193855.58	194171.80	0.08	**193698.00**	193978.10	0.00
G9(255)	14	4723.77	4730.70	4740.42	0.15	4737.32	4744.14	0.29	4725.58	4728.05	0.04	**4723.77**	4727.45	0.00	4723.95	4731.05	0.00
G10(323)	16	6712.53	6732.36	6747.10	0.30	6721.16	6742.76	0.13	6713.92	6717.76	0.02	6714.94	6719.24	0.04	**6712.53**	6724.70	0.00
G11(399)	18	9210.45	9243.05	9259.66	0.35	9240.37	9257.23	0.32	9214.07	9216.60	0.04	9217.63	9225.56	0.08	**9210.45**	9228.90	0.00
G12(483)	19	12507.50	12629.37	12649.21	0.97	12659.15	12708.54	1.21	12526.17	12543.04	0.15	12549.73	12570.26	0.34	**12507.50**	12567.80	0.00
G13(252)	26	3628.30	3653.07	3660.93	0.68	3686.62	3965.98	1.61	**3628.30**	3638.50	0.00	3635.44	3656.13	0.20	3632.61	3658.95	0.12
G14(320)	29	5205.75	5770.02	6045.20	10.83	5826.43	6420.08	11.91	5216.80	5257.95	0.20	5290.05	5309.65	1.61	**5205.75**	5218.60	0.00
G15(396)	33	7010.13	7077.48	7140.11	0.96	8576.61	8576.61	22.35	7010.41	7023.12	0.00	7026.96	7075.33	0.24	**7010.13**	7028.93	0.00
G16(480)	37	9248.37	9300.74	9339.45	0.57	9347.02	9420.36	1.07	9250.98	9268.30	0.03	9271.40	9293.68	0.25	**9248.37**	9276.40	0.00
G17(240)	22	3063.02	3089.99	3130.99	0.87	3095.32	3117.79	1.04	3065.46	3068.29	0.07	3064.47	3073.96	0.03	**3063.02**	3068.41	0.00
G18(300)	27	4216.01	4528.16	4582.44	7.34	5083.70	5138.01	20.51	4221.14	4244.60	0.06	4311.06	4352.10	2.19	**4216.01**	4235.92	0.00
G19(360)	33	5502.08	5570.35	5589.12	1.24	5600.05	5631.68	1.78	5523.38	5531.78	0.39	5513.72	5534.05	0.21	**5502.08**	5537.58	0.00
G20(420)	38	7209.31	7413.58	7473.69	2.83	8453.96	8453.96	17.26	7223.08	7240.86	0.19	7237.40	7303.18	0.39	**7209.31**	7239.89	0.00
Average			65072.07	65314.68	1.53	65594.26	65824.53	4.35	64979.25	65213.33	0.22	64952.50	65072.10	0.42	**64707.39**	64804.86	**0.01**
#Best			2			1			1			4			**17**		
#Worst			3			13			5			0			0		
$Gap_{robustness}$ (%)			0.66			1.20			0.26			0.30			0.26		

In Set 2, we obtain 15 new best known results out of 20 instances as shown in Table 3. These include G2–G4, G7–G8, G10–G12, G14–G20 instances. Besides, two of our other solutions (G5–G6) match the best known values whereas the remaining three are very close to the best known results. The largest solution improvement is observed for instance G7 where 0.61% reduction over the previous best solution (Ribeiro and Laporte, 2012) is recorded. On the other hand, the worst performance are instances G1 and G13 where a 0.12% deterioration is observed. In addition, the average percentage deviation obtained by our algorithm is less than a tiny 0.01%, which is the smallest among all the existing CCVRP algorithms in the literature. In terms of the relative performance, our algorithm is also one of the most robust, as shown by a small gap of 0.26% between the average and the best solution.

Table 4: Computational time for Set 1a (Christofides et al., 1979) instances.

$\#(n)$	$\lvert R\rvert$	T_{Avg} (s)				
		MA1	MA2	ALNS	2-Meta	AVNS
C1(50)	5	10.63	3.70	30.29	14.70	11.89
C2(75)	10	27.78	2.04	60.77	18.73	15.04
C3(100)	8	97.91	40.46	172.45	50.80	41.03
C4(150)	12	449.44	188.41	235.12	77.23	60.54
C5(199)	17	1035.45	629.27	277.37	112.80	89.75
C11(120)	7	160.64	68.83	202.07	59.70	48.32
C12(100)	10	38.20	23.66	152.74	44.77	34.21
Average		260.01	136.62	161.54	54.10	42.97
Comp spec		3.4GHz	3.4GHz	2.0GHz	2.4GHz	3.4GHz

Table 4 and Table 5 record the corresponding average computational time over 5 runs (T_{Avg} in seconds) for different algorithms. On average, the best solutions in Set 1a are obtained within 42.97 seconds, and within 570.62 seconds for Set 2. While most of the existing algorithms (MA and ALNS) are quite computational intensive, the AVNS alongside the two-phase metaheuristic are very competitive.

Table 6 presents the new results for Set 1b. Since these instances have not been tested by any other algorithm for the CCVRP, we report the average and the best solution value, the gap between these two as well as the average computational time. The gap between the average and the best solution for Set 1b is 0.48%, which is the greatest among all the four data sets tested but it is still acceptable. The computational time is rather small, with just about 8.99 seconds on average.

In addition, the results on the large data set (Set 3) are also shown in Table 7. It can be observed that the time spent on the largest instance with 1200 customers (L12) is about 2.9 times the time spent on the L1 instance of 560 customers. The robustness behaviour of our algorithm has shown again to be strong given its tiny gap of 0.29% between the best and the average solution.

Table 5: Computational time for Set 2 (Golden et al., 1998) instances.

#(n)	|R|	T_{Avg} (s)				
		MA1	MA2	ALNS	2-Meta	AVNS
G1(240)	9	1593.60	866.80	1038.27	289.69	252.51
G2(320)	10	4549.05	2054.68	1484.74	707.13	710.65
G3(400)	10	9295.64	3336.09	2061.75	1222.34	1217.24
G4(480)	10	12810.31	6557.67	2626.89	1622.14	1484.21
G5(200)	5	471.45	246.05	1200.67	291.52	248.45
G6(280)	7	2358.40	788.71	1547.43	456.82	478.60
G7(360)	8	1537.36	258.45	1926.19	357.77	660.21
G8(440)	10	9598.42	5165.83	2330.34	1244.40	1089.35
G9(255)	14	2453.61	1582.86	864.89	258.62	216.58
G10(323)	16	7652.00	4332.24	1092.34	403.54	441.65
G11(399)	18	8835.74	7429.60	1356.99	590.97	573.07
G12(483)	19	9881.24	3449.06	1540.32	691.26	626.36
G13(252)	26	2012.17	74.31	632.72	163.07	125.93
G14(320)	29	2263.84	9.59	682.19	132.36	161.77
G15(396)	33	5597.98	0.00	855.25	665.37	604.68
G16(480)	37	16204.52	5307.29	1104.53	1015.31	944.18
G17(240)	22	1631.86	431.31	618.70	287.68	237.98
G18(300)	27	2410.30	0.10	630.47	337.73	344.73
G19(360)	33	5535.25	763.56	853.43	525.52	465.40
G20(420)	38	6264.59	0.00	881.92	604.83	528.79
Average		5647.87	2132.71	1266.50	593.40	570.62

Table 6: New results for Set 1b (Fisher, 1994) instances.

#(n)	|R|	AVNS		Gap (%)	T_{Avg} (s)
		C_{Best}	C_{Avg}		
F1(44)	10	2411.82	2428.52	0.69	4.34
F2(71)	14	1781.86	1789.54	0.43	8.04
F3(134)	10	6449.04	6469.85	0.32	14.60
Average		3547.57	3562.64	0.48	8.99

5. The CCVRP with min-max objective

An alternative objective that minimises the latest arrival time instead is also investigated in this study. As the cumulative effect is represented by the last customer in the longest route (i.e., the arrival times are accumulated from the previous customers), this problem reduces to the min-max open VRP (OVRP). For convenience and to keep within the framework of cumulation, we refer to this problem as the min-max CCVRP though the min-max OVRP could also be used.

Let T be the latest arrival time at a customer, the min-max objective function is then stated in Equation (18) where $T \geq t_i^k$ for all $i \in V'$ and $k \in R$. The constraints of this problem are similar to the ones in the CCVRP given by Ngueveu et al. (2010).

$$\min C(x) = T \tag{18}$$

Table 7: New results for Set 3 (Li et al., 2005) instances.

#(n)	|R|	AVNS		Gap (%)	T_{Avg} (s)
		C_{Best}	C_{Avg}		
L1(560)	10	374376.00	374875.60	0.13	1209.00
L2(600)	14	218301.00	219345.40	0.48	1491.60
L3(640)	10	506252.00	507575.80	0.26	1406.40
L4(720)	10	659440.00	661735.62	0.35	1599.00
L5(760)	17	251756.00	252297.30	0.22	2132.40
L6(800)	10	835039.00	836743.50	0.20	1932.60
L7(840)	19	263303.00	264943.60	0.62	2258.40
L8(880)	10	1030330.00	1031806.50	0.14	2379.00
L9(960)	10	1243250.00	1251148.90	0.64	2594.40
L10(1040)	10	1479510.00	1481868.30	0.16	2740.20
L11(1120)	10	1736150.00	1738320.43	0.13	3084.60
L12(1200)	10	2013670.00	2016517.30	0.14	3459.00
Average		884281.42	886431.52	0.29	2190.55

Differences between the min-sum CCVRP, min-max CCVRP and the min-max VRP

The min-max objective aims to minimise the route with the largest cost. However, the cost of a route is defined differently for the min-sum, the min-max CCVRP and the VRP as follows:

(a) min-sum CCVRP – the total cumulative distances (i.e., sum of arrival times) at the customers excluding the last vertex (the depot),

(b) min-max CCVRP – the maximum distance travelled (i.e., the arrival time at the last customer) excluding the return trip from the last customer to the depot,

(c) min-max VRP – the maximum distance travelled for the complete round-trip (i.e., the arrival time at the depot).

It is worth noting that the min-max CCVRP is different from the min-max VRP in terms of the inclusion of the last edge in the route. In other words, the former aims to reduce the cumulative distance at the last customer whereas the latter intends to minimise the total distance of the longest route including the return trip from the last customer back to the depot. In the context of humanitarian relief, it is more important to deliver aid supplies promptly to the victims than sending the vehicles back to the depot quickly.

Since the return trip to the depot is excluded, the min-max CCVRP can also be seen as the counterpart of the min-max open VRP, although the number of vehicles is predefined for the former. In addition, the open routes (the min-max CCVRP) instead of the closed ones (the min-max VRP) may affect the solutions to some extent.

To better illustrate the difference between the min-max CCVRP and the min-max VRP, an example is provided in Figure 3. Here, route R_2 is the longest for the former, and route R_1

for the latter. Therefore, the min-max CCVRP intends to reduce the arrival time at the last customer (customer 'b') in route R_2.

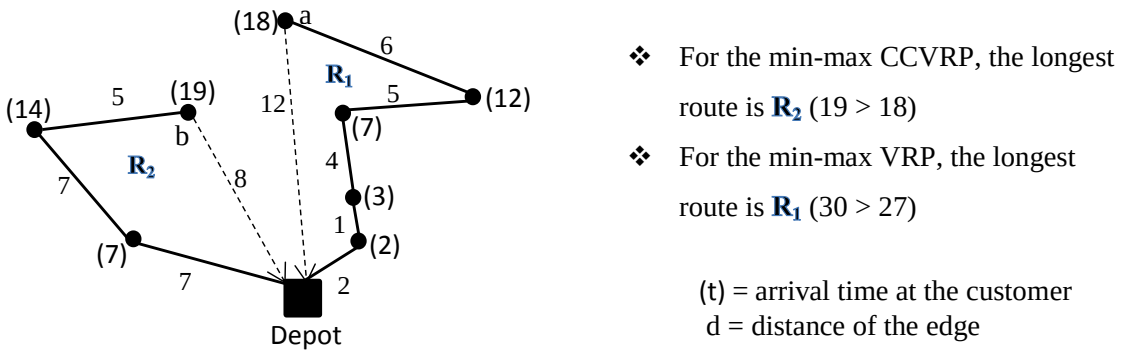

Figure 3: An illustration example highlighting the difference between the min-max CCVRP and the min-max VRP.

Note that another version of the min-max CCVRP would be the minimisation of the route with the largest sum of arrival times (i.e., the largest cost for the min-sum CCVRP). However in practice, it is more meaningful to minimise the arrival time at the last customer as defined in (b) than the route with the largest total cumulative cost as in (a). For example in Figure 3, for the case of the CCVRP (min-sum), the longest route is R_1 (sum of arrival times for $R_1 = 42$ versus 40 for route R_2). If the aforementioned version is adopted, route R_1 will be chosen. Nevertheless, the vehicle arrival time at the last customer in route R_1 (customer 'a') is 18, which is smaller than the arrival time of the last customer in route R_2 (arrival time at customer 'b' = 19). This implies that customer 'b' actually waits longer than customer 'a'. In terms of practicality, the min-max CCVRP considered in this paper aims to minimise the arrival time of the vehicle at the last customer, which is consistent with the one discussed by Campbell et al. (2008).

5.1. Adaptations of the AVNS algorithm

As we aim to minimise the latest arrival time which is governed by the longest route, some strategies that capture this characteristic are proposed in the following subsections: (i) the heuristic method for the construction of an initial solution, (ii) the move evaluation process and (iii) the necessary adaptations of the local search engine used in the AVNS algorithm.

5.1.1. Heuristic for the initial solution

A similar greedy insertion method, as discussed in Section 3.2, is also employed here to construct the initial solution. For customer i, we first determine if this customer can be inserted without an increase in the current objective function value. If there is more than one option available, the move that causes the smallest increment to the maximum arrival time of the associated route

is selected. If there is no such move, the customer will be assigned to the route that has the smallest increment in the objective function value.

To achieve this, the cost of insertion of customer i is defined as the difference between the new maximum arrival time $f_{i,k}$ after customer i is inserted into route k, and the current objective value, $C(x)$ as follows: $\Delta f_{i,k} = f_{i,k} - C(x)$, where $k \in R$. The customer with the smallest $\Delta f_{i,k}$ (which could also be negative) is inserted each time, until all customers are assigned to the routes.

5.1.2. Evaluation of the cost of a move

The cost variations of removal and insertion are computed in a similar way as in the classical VRP, except that the last edge of the route is not considered since the vehicle arrival time at the depot is not included in the objective function. For example, consider the set of customers in route k (R_k^0, R_k^1,..., $R_k^{n_k}$, $R_k^{n_k+1}$) with R_k^0 and $R_k^{n_k+1}$ being the depot. To remove u customer(s) starting from the i^{th} to the $(i+u-1)^{th}$ customer from the route, the cost variation of removing, $\psi_k^-(u)$ is given in Equation (19). Note that this equation is similar to $\Delta G^+(k,u,i)$ in Equation (11) except that the distance of the last edge in the route is set to be zero.

$$\psi_k^-(u) = \sum_{a=R_k^{i-1}}^{R_k^{i+u-1}} d_{a,a+1} - d_{R_k^{i-1},R_k^{i+u}}, \qquad \text{if } R_k^{i+u} = R_k^{n_k+1}, d_{R_k^{i-1},R_k^{i+u}} = d_{R_k^{i+u-1},R_k^{i+u}} = 0 \tag{19}$$

Similarly for insertion, if a customer is to be placed at the last position of a route, the last edge is omitted. Using the same route in the example above, to insert u customers which is being removed from route k at position j in route l, the cost variation of insertion, $\psi_l^+(u)$ is stated in Equation (20).

$$\psi_l^+(u) = d_{R_l^{j-1},R_l^j} - (d_{R_l^{j-1},R_k^i} + \sum_{a=R_k^i}^{R_k^{i+u-2}} d_{a,a+1} + d_{R_k^{i+u-1},R_l^j}), \text{if } R_l^j = R_l^{n_l+1}, d_{R_l^{j-1},R_l^j} = d_{R_k^{i+u-1},R_l^j} = 0 \tag{20}$$

5.1.3. Adaptations of the AVNS local search engine

The following three in the AVNS algorithm, as discussed in Section 2, is modified: (i) the parameter used to define the neighbourhood reduction, (ii) the shaking step, and (iii) the local search step. As our aim here is to reduce the longest route, we allow more flexibility to shift customers from the longest route to the shorter ones. To achieve this, in (i), the q^{th} value of the neighbourhood reduction in Section 3.3 is increased from 3% to 8%.

Moreover, in (ii), the shaking is performed on the longest route only. In other words, in the

inter-route neighbourhood structures, we aim to remove some customers from the longest route and relocate them to other routes which are sorted in increasing order of the route cost. The algorithm first considers shifting customer(s) to the shortest route, and if there is no feasible move then the second shortest route is checked and so on. Note that the routes can also be sorted according to the route capacity instead of the route cost. In some cases, the route may have a high capacity but the cost is relatively low. The customers can be inserted in such routes that remain within a certain violation as discussed in Section 3.4.

In (iii), we define the local search engine for both stages of the AVNS algorithm in two ways: (1) Basic Search (BS) and (2) Extended Search (ES). More specifically, the BS only evaluates moves between the longest route against the other routes whereas the ES examines all routes in the solution. The idea is when the longest route cannot be further reduced, it is worth refining the other routes, with the aim that the longest one could have more chances for improvement at a later point given the changes in the configuration of the other routes. The steps of the local search in the AVNS Stage 1 are presented in Algorithm 2.

In brief, firstly the BS evaluates and performs moves for the longest route, l and against other routes until there is no more improvement (lines 3–8 in Algorithm 2). The ES is then activated by exploring all routes. Here, we define the gain of a move in two versions: (a) when one of the routes considered is the longest route, and (b) when none of the routes is the longest. For (a), the gain is defined as $Gain(L_h) = \max_{l \in R}\{\Delta C_l(u, w)\}$ (lines 13–14) where $\Delta C_l(u, w)$ is the gain in route l due to an operator that removes u customer(s) from, and inserts w customer(s) into route l (see Section 3.5.3 for details). For (b), the evaluation of the cost of a move is based on the total gains obtained in the non-longest routes a and b, which is given as: $Gain(L_h) = \max_{a,b \in R \setminus \{l\}}\{\Delta C_a(u, w) + \Delta C_b(w, u)\}$ (lines 15–16). This type of move cannot reduce the objective value, but it can improve the arrival time at the last customer in the non-longest routes. Therefore, customers from the longest route may now have more flexibility to be inserted to these routes.

Similarly, both the BS and the ES are also implemented in the multi-level local search engine in Stage 2 of the AVNS as shown in Algorithm 3. The algorithm first evaluates if there is improvement on the longest route (lines 4–11 in Algorithm 3), and then on the other routes (lines 16–22). It is worth mentioning that when evaluating the moves in both Stage 1 and Stage 2, any move that increases the current objective value is discarded from further examination, leading to an effective and guided search.

5.2. Computational experience for the min-max CCVRP

The adapted AVNS algorithm is run on Set 1–Set 3 and the results are presented in Table 8. For consistency, the AVNS algorithm is executed over 5 runs using the same stopping criterion as previously defined. The best results obtained together with the average time are also reported

Algorithm 2 The *best improvement local search* in the AVNS algorithm (Stage 1): case of the min-max CCVRP.

Define the longest route, l.
(1) Basic Search (BS):
repeat
  for ($h = 1$ to $h = h_{max}$) **do**
    Search in route l and against other routes.
  end for
  Select the maximum gain, $\max_{1 \leq g \leq h_{max}} Gain(L_g)$ and perform the move to obtain x''.
until (maximum gain ≤ 0)
(2) Extended Search (ES):
repeat
  for ($h = 1$ to $h = h_{max}$) **do**
    Evaluate moves using operator L_h on all routes.
    if (one of the routes is l) **then**
      $Gain(L_h) = \max_{l \in R}\{\Delta C_l(u, w)\}$.
    else
      $Gain(L_h) = \max_{a,b \in R \setminus \{l\}}\{\Delta C_a(u, w) + \Delta C_b(w, u)\}$.
    end if
  end for
  Check $Gain(L_h)$ recorded for route l.
  if ($Gain(L_h)$ for route $l > 0$) **then**
    Select the maximum gain in route l.
  else if ($Gain(L_h)$ for the other routes > 0) **then**
    Select the maximum gain.
  end if
  Perform the move to obtain x''
until (maximum gain ≤ 0)

in this table. Here, we restrict the number of routes for each instance to be the same as the ones used in the CCVRP.

5.2.1. The results for the min-max CCVRP

To the best of our knowledge, there are no results published in the literature for this problem. Performance measures to evaluate the quality of our solution exist, some of which include: (i) lower and upper bounds as in the study of Lysgaard and Wøhlk (2014) for the CCVRP; (ii) tight upper bounds obtained by other powerful and effective metaheuristics; and (iii) derivation of the corresponding solution values from an already known solution of a related problem. In (i), useful information could be found for small size instances whereas in (ii), a good quality solution could be obtained. For instance in (iii), the min-sum CCVRP solution can be converted to the maximum arrival time. This is achieved by retrieving the arrival time at the last customer in the longest route from the min-sum solution. Obviously these corresponding objective function values would be inferior but can still be used as a basis for comparison purposes as highlighted

Algorithm 3 The *multi-level* K^{th} *best improvement local search* in the AVNS algorithm (Stage 2): case of the min-max CCVRP.

1: Define the longest route, l.
2: **(1) Basic Search (BS):**
3: **while** $(m < m_{max})$ **do**
4: Evaluate moves using operator W_m on the longest route l and against other routes with:
5: $Gain(W_m) = \underset{l \in R}{\text{Kmax}}\{\Delta C_l(u, w)\}$.
6: **if** $(Gain(W_m) < 0)$ **then**
7: $m \leftarrow m + 1$.
8: **else**
9: Perform the K^{th} best improvement move to obtain x''.
10: $m \leftarrow 1$.
11: **end if**
12: **end while**
13: **(2) Extended Search (ES):**
14: **if** (no improvement move is found for all operators) **then**
15: **while** $(m < m_{max})$ **do**
16: Evaluate moves using operator W_m on all routes.
17: **if** (one of the routes is l) **then**
18: $Gain(W_m) = \underset{l \in R}{\text{Kmax}}\{\Delta C_l(u, w)\}$.
19: **else**
20: $Gain(W_m) = \underset{a,b \in R \setminus \{l\}}{\text{Kmax}}\{\Delta C_a(u, w) + \Delta C_b(w, u)\}$.
21: **end if**
22: Same as lines 6–11.
23: **end while**
24: **end if**

in Campbell et al. (2008).

Measures (i) and (ii) are interesting but fall beyond the scope of this study. Here we opt for alternative (iii) as the results for the min-sum CCVRP solution configurations are already available to us (see Section 4.2). The best results of the min-max CCVRP and the converted values of the maximum arrival time obtained from the CCVRP best solutions are given in Table 8.

The result shows that the maximum arrival time for the min-max CCVRP is 7.21% better than the value converted from the min-sum CCVRP. Besides this improvement being positive as one may expect, this also demonstrates that it is better to solve the 'true' problem directly instead of its counterpart. Furthermore, the average CPU time for the min-max is slightly higher than the one of min-sum problem although the average deviation indicates that the former is about 7.55% shorter than the latter. The conversion time of the solution is negligible and hence not reported here.

It is also interesting to observe that the solutions for G2–G6 in Set 2, L3 and L9 in Set 3 have the same values for both the min-max and the min-sum objectives. These values are underlined

in Table 8. Since these instances establish a concentric circle structure, all the routes found have the same structure as shown in Figure 4 where the solutions for G2–G4 are plotted. A further evaluation of the route configurations shows that the solutions obtained for these instances are nearly optimal. Here, each route has an equal number of customers and the same cost.

Table 8: New results for the min-max CCVRP and the converted best results of the min-sum CCVRP.

	Min-max (MM)		Min-sum (MS)$_{best}$			Min-max (MM)		Min-sum (MS)$_{best}$	
$\#(n)$	ma	T_{Avg} (s)	ma	T^{+}_{Avg}(s)	$\#(n)$	ma	T_{Avg} (s)	ma	T^{+}_{Avg}(s)
C1(50)	90.20	9.85	105.72	11.89	G11(399)	40.79	559.41	43.46	573.07
C2(75)	62.96	12.17	67.63	15.04	G12(483)	46.72	605.10	47.70	626.36
C3(100)	85.55	34.41	110.46	41.03	G13(252)	25.10	94.65	27.83	125.93
C4(150)	65.00	57.12	70.85	60.54	G14(320)	29.20	151.32	29.93	161.77
C5(199)	56.42	92.32	60.27	89.75	G15(396)	31.14	462.70	32.32	604.68
C11(120)	118.96	44.10	141.29	48.32	G16(480)	33.96	848.21	35.43	944.18
C12(100)	63.88	32.14	78.26	34.21	G17(240)	22.92	186.10	25.70	237.98
F1(44)	147.66	4.54	161.30	4.34	G18(300)	27.24	198.49	30.66	344.73
F2(71)	49.94	7.04	61.92	8.04	G19(360)	31.70	385.82	35.84	465.40
F3(134)	152.66	14.65	182.70	14.60	G20(420)	38.68	440.68	45.45	528.79
G1(240)	523.65	219.65	528.36	252.51	L1(560)	1514.21	1138.60	1570.70	1209.00
G2(320)	734.85	713.95	734.85	710.65	L2(600)	871.84	1488.30	887.21	1491.60
G3(400)	993.68	1321.65	993.68	1217.24	L3(640)	1770.17	1431.60	1770.17	1406.40
G4(480)	1252.51	1504.20	1252.51	1484.21	L4(720)	2031.87	1627.54	2029.00	1599.00
G5(200)	1234.21	247.78	1234.21	248.45	L5(760)	778.50	2134.64	798.38	2132.40
G6(280)	1119.48	485.87	1119.48	478.60	L6(800)	2304.52	2178.60	2402.48	1932.60
G7(360)	1171.67	566.95	1210.18	660.21	L7(840)	726.65	2794.80	748.86	2258.40
G8(440)	1047.27	1075.65	1057.85	1089.35	L8(880)	2558.00	2608.90	2652.07	2379.00
G9(255)	32.48	190.28	34.66	216.58	L9(960)	2805.49	2665.20	2805.49	2594.40
G10(323)	36.49	424.83	38.65	441.65	L10(1040)	3065.03	2910.94	3064.32	2740.20
					L11(1120)	3324.86	3204.81	3323.15	3084.60
					L12(1200)	3590.37	3647.84	3630.81	3795.00
Average						826.39	924.37	842.60	913.40
Dev^{ma,T}(%)								7.21	7.55

ma: maximum arrival time, T^{+}_{Avg}: average computational time over 5 runs for the min-sum CCVRP where the conversion time is negligible, underline means both the min-max and the min-sum objective values are the same.
Dev^{b} = $(b(\text{MS}_{best}) - b(\text{MM}))/b(\text{MM}) \times 100$, $b = \{ma, T\}$.

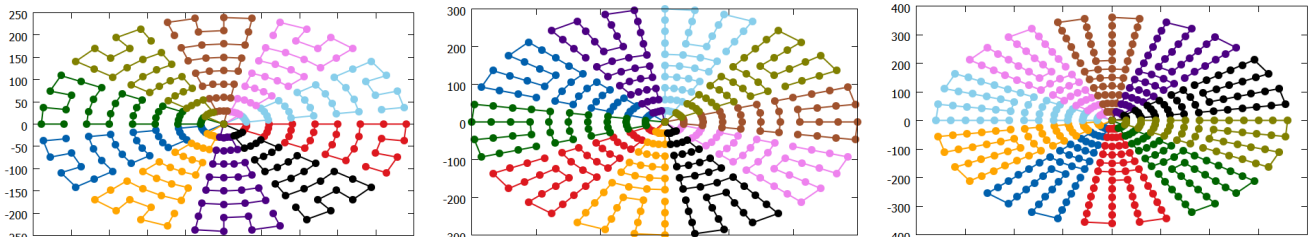

Figure 4: Graphical representations of the solutions for instances G2(320), G3(400) and G4(480).

For completeness, a similar table that shows the sum of arrival times for the min-sum CCVRP and the converted best results from the min-max CCVRP is also presented (see Table 9). The computational times are omitted here as they have been previously displayed in Table 8. One can see that the solutions for the min-sum CCVRP are, on average, 3.94% better than the ones for the min-max CCVRP.

Table 9: The sum of arrival times for the min-sum CCVRP and the converted best results of the min-max CCVRP.

$\#(n)$	sa(MS)	sa(MM$_{best}$)	$\#(n)$	sa(MS)	sa(MM$_{best}$)	$\#(n)$	sa(MS)	sa(MM$_{best}$)
C1(50)	2230.35	2416.82	G5(200)	114163.64	114163.64	G19(360)	5502.08	5889.51
C2(75)	2391.63	2469.83	G6(280)	140430.09	140430.09	G20(420)	7209.31	8199.46
C3(100)	4045.42	4234.24	G7(360)	179388.00	182658.00	L1(560)	374376.00	378482.00
C4(150)	4987.52	5131.48	G8(440)	193698.00	195370.00	L2(600)	218301.00	221163.00
C5(199)	5806.02	6104.95	G9(255)	4723.95	4815.50	L3(640)	506252.00	506252.00
C11(120)	7314.55	7812.06	G10(323)	6712.53	6867.99	L4(720)	659440.00	660294.00
C12(100)	3558.92	3949.35	G11(399)	9210.45	9577.25	L5(760)	251756.00	255167.00
F1(44)	2411.82	3147.09	G12(483)	12507.50	13238.20	L6(800)	835039.00	850949.00
F2(71)	1781.86	1956.22	G13(252)	3632.61	3720.00	L7(840)	263303.00	265799.00
F3(134)	6449.04	7512.79	G14(320)	5205.75	5449.60	L8(880)	1030330.00	1033370.00
G1(240)	54749.30	55353.10	G15(396)	7010.13	7155.76	L9(960)	1243250.00	1243250.00
G2(320)	100560.00	100560.00	G16(480)	9248.37	9707.84	L10(1040)	1479510.00	1481930.00
G3(400)	170924.00	170924.00	G17(240)	3063.02	3123.59	L11(1120)	1736150.00	1737590.00
G4(480)	261993.00	261993.00	G18(300)	4216.01	4434.47	L12(1200)	2013670.00	2017250.00
Average							284440.52	285657.02
Dev^{sa} (%)								3.94

sa(MS): sum of arrival times for min-sum CCVRP, sa(MM$_{best}$): converted sum of arrival times for min-max CCVRP, underline means both the min-sum and the min-max objective values are the same.
Dev^{sa} = (sa(MM$_{best}$) − sa(MS))/sa(MS) × 100.

5.2.2. The effect of the Extended Search (ES)

To evaluate the effect of the ES, the AVNS algorithm with and without the ES and using the same stopping criterion, as defined in Section 4.2, is tested. In other words, the local search engine only evaluates improvement moves on the longest routes and performs the moves until the stopping criterion is reached (i.e., the BS only) when the ES is omitted. With the use of the ES, it is observed that the solutions in Set 1–Set 3, improved on average by 3.08% while consuming about 20.79% extra computational time. Interestingly, the sum of arrival times of the min-max solution is reduced by 24.36% when the ES is considered. Note that for the min-max objective, it is possible to have many solutions under the same maximum value. This empirical experiment suggests that the ES could be an effective mechanism in improving the solution in terms of the maximum arrival time as well as ensuring a reasonable total arrival times at the customers.

6. Managerial insights of using different objective functions for decision making

To compare the solutions under different objective functions, we construct the bar charts for the solutions obtained using the min-sum and the min-max alongside with the classical VRP objective as shown in Figure 5. Note that the classical VRP solutions are based on the results obtained in Sze et al. (2016), which are shown to be competitive with the BKS published in the literature besides the details of the routes are available to us for conversion purpose. The notations used include sa (sum of arrival times), ma (maximum arrival time) and d (distance) which correspond to the objective function values of the min-sum, min-max CCVRP and VRP

respectively. Note that the VRP solutions for some instances are not comparable with the CCVRP and the min-max CCVRP in terms of the number of routes and the presence of the route length constraint, and therefore are omitted from the graphs. These instances are G1–G8, G14–G15 from Set 2 and all instances in Set 3.

It is clear that the solution of a particular problem based on its own objective function should be better. Moreover, the differences between the min-sum and the min-max objectives are relatively small compared to the VRP, for all three values of sa, ma and d. Besides, one can note that the VRP solution has significantly higher sa and ma values. This implies that the VRP objective is not suitable in the context when the arrival times at the customers is crucial such as the humanitarian logistic. On average, the min-sum solution is 7.41% better in terms of the sa and 0.84% lower in the d than the min-max whereas the min-max is 13.07% improved in the ma compared to the min-sum. Furthermore, there is a significant gap between the VRP total distance and the min-sum and the min-max solutions (25.23% and 26.21% respectively).

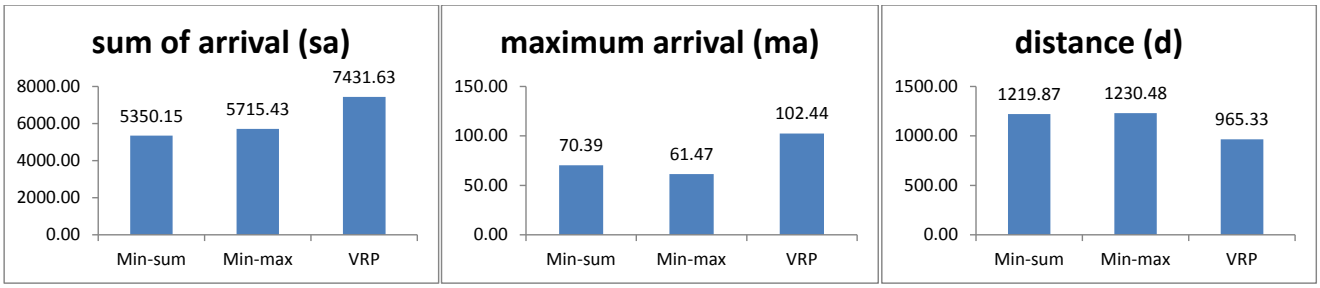

Figure 5: Comparing the min-sum, the min-max and the VRP objective function values.

While it is clear that the VRP solution appears to be relatively poor in the min-sum and the min-max contexts, the differences between the min-sum and the min-max CCVRP in terms of the sa and ma values may be considered small but can be very important from a managerial perspective. Therefore to further evaluate the trade-off between the improvements in the sum of arrival times and the maximum arrival time using the two objectives, we plot the points $(x, y) = (\frac{sa(\text{MM})-sa(\text{MS})}{sa(\text{MS})}, \frac{ma(\text{MS})-ma(\text{MM})}{ma(\text{MM})})$ for Set 1–Set 3, as shown in Figure 6. For example, the notation $sa(\text{MM})$ represents the sum of arrival times when using the min-max objective. This plot shows the percentage increase in the sum of arrival times using the min-max objective against the percentage increase in the maximum arrival time using the min-sum objective. Since most of the points lie above the line $y = x$, it is reasonable to conclude that the relative increase in the maximum arrival time is more significant when the min-sum objective is used than the relative increase in the sum of arrival times when the min-max objective is applied. It appears that the min-max objective is able to obtain high quality solutions in terms of the maximum arrival time as well as achieving a satisfactory total arrival times. One reason for this is the use of the extended search when solving the min-max CCVRP that also reduces the total arrival times.

Let $\tilde{t}_i$ be the arrival time of a vehicle serving customer i, where $i \in V'$. To analyse the arrival

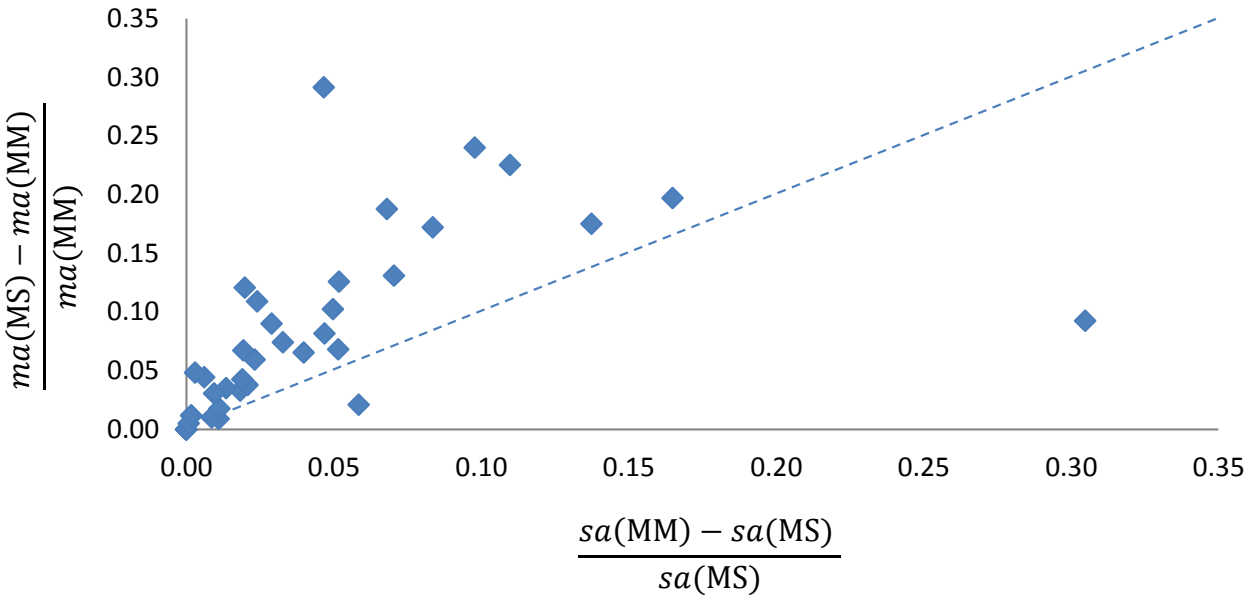

Figure 6: Improvement in the sum of arrival times versus the maximum arrival time.

time at individual customer in the min-sum and the min-max solutions, we compute the standard deviation, σ in Equation (21). A solution with a lower standard deviation tends to have more balanced arrival times among the customers.

$$\sigma = \sqrt{\frac{\sum(\tilde{t}_i - \bar{t})^2}{n-1}}, \qquad \text{where } \bar{t} = \frac{\sum_{i \in V'} \tilde{t}_i}{n} \tag{21}$$

The minimum, average and maximum ratio of $\sigma(\text{MS})/\sigma(\text{MM})$ are presented in Table 10. For example, the minimum ratio of Set 1a is computed by the minimum value of $\sigma(\text{MS})/\sigma(\text{MM})$ in that set whereas the average ratio weights each of the instances equally which provides a simple comparison among the different data sets. Here, the further this ratio deviates from 1 the greater the objective function changes the individual customer's standard deviation in the solutions.

It is observed that the ratios are smaller than 1 for most instances except for the maximum ratios in Set 2 and Set 3, where the values are slightly greater than 1. This suggests that the CCVRP min-sum objective is able to balance the overall individual customer's arrival time compared to the min-max objective. Another observation made is that when the customer points exhibit a geometric structure (i.e., concentric circles, squares and stars in Set 2 and Set 3), the solutions for both the min-sum and the min-max appear to be similar. In managerial decision making, the min-sum would be preferable if one aims to reduce the overall customers arrival times. On the other hand, if the situation requires that the last customer must be served before the deadline, especially for the case where there is a high penalty for lateness, the min-max objective would provide promising and practical result.

Table 10: The standard deviation ratio for the min-sum and the min-max CCVRP.

		$\frac{\sigma(\text{MS})}{\sigma(\text{MM})}$			$\frac{\sigma(\text{MS})}{\sigma(\text{MM})}$			$\frac{\sigma(\text{MS})}{\sigma(\text{MM})}$			$\frac{\sigma(\text{MS})}{\sigma(\text{MM})}$
Set 1a	Min	0.804	Set 1b	Min	0.736	Set 2	Min	0.911	Set 3	Min	0.989
	Avg	0.839		Avg	0.848		Avg	0.978		Avg	0.999
	Max	0.929		Max	0.910		Max	1.004		Max	1.006

7. The effects of doubling up the number of runs

Our experiments for the min-sum and the min-max CCVRP in Section 4 and Section 5.2 respectively are executed using 5 independent runs of the AVNS algorithm as this is used in the literature. In this section, we conducted the same experiments to evaluate the effect of a larger number of runs on the solutions. The results for the min-sum and the min-max CCVRP are presented in Table 11 and Table 12 respectively where the best values (sa_{Best} and ma_{Best}), average values (sa_{Avg} and ma_{Avg}) together with the average time (T_{Avg}) are shown.

It can be observed that there is not much difference between the results for 5 and 10 runs for the min-sum CCVRP. More specifically, the deviations are just about 0.01% between 5 and 10 runs for the min-sum sa_{Best} and 0.03% for the sa_{Avg}. For Set 1a and Set 1b, both 5 and 10 runs achieve the same sa_{Best} for all instances, whereas in Set 2, 7 better results are found for 10 runs and in Set 3, 9 new results are obtained. These new results are underlined in Table 11. Note that all the values provided in Table 11 are the BKS except for instances G1, G9 and G13 (marked by a '+') where better results are reported in Table 3.

Similarly, the difference between the results of 5 and 10 runs for the min-max CCVRP is also relatively tiny. As Table 12 shows, the gap is 0.06% for the ma_{Best} and 0.09% for the ma_{Avg} between 5 and 10 runs. To be specific, all the ma_{Best} are the same for Set 1a and Set 1b, 7 better results are obtained for Set 2 and 6 for Set 3 by using 10 runs as compared to 5 runs. It is worth mentioning for some instances although the maximum arrival time (ma_{Best}) is the same for both 5 and 10 runs, the average gap for the sum of arrival times is about 0.66% between the two sets of runs. The small deviation between 5 and 10 runs indicate the robustness of our algorithm in achieving consistent and high quality results even for a small number of runs.

8. Conclusion

The CCVRP is a relatively new variant of the classical VRP with useful applications particularly in humanitarian logistic. In this study, an effective AVNS algorithm which incorporates a LNS diversification strategy is proposed to solve the CCVRP. The prominent features of the algorithm include: (i) the intelligent and effective selection of local search operators, (ii) an efficient cost evaluation scheme of moves, (iii) the use of the LNS diversification strategy, and (iv) its robustness and easiness to use and to modify. The proposed algorithm besides being robust is proven to

Table 11: Results for the min-sum CCVRP using 10 runs.

$\#(n)$	sa_{Best}	sa_{Avg}	T_{Avg} (s)	$\#(n)$	sa_{Best}	sa_{Avg}	T_{Avg} (s)
C1(50)	2230.35	2242.47	11.70	G11(399)	9210.45	9230.59	583.79
C2(75)	2391.63	2392.54	14.98	G12(483)	12506.40	12550.20	614.57
C3(100)	4045.42	4068.71	41.76	G13(252)	$^{+}$3632.61	3654.13	123.90
C4(150)	4987.52	4991.65	61.47	G14(320)	5205.75	5216.86	157.42
C5(199)	5806.02	5828.57	86.09	G15(396)	7010.13	7031.42	609.84
C11(120)	7314.55	7330.97	46.62	G16(480)	9243.35	9269.51	964.59
C12(100)	3558.92	3560.18	35.15	G17(240)	3063.02	3066.02	234.20
F1(44)	2411.82	2417.62	4.69	G18(300)	4216.01	4237.85	363.40
F2(71)	1781.86	1797.53	8.83	G19(360)	5501.21	5534.90	451.73
F3(134)	6449.04	6460.75	15.08	G20(420)	7207.88	7228.96	520.98
G1(240)	$^{+}$54739.80	54778.90	248.34	L1(560)	374285.00	374865.40	1269.20
G2(320)	100560.00	100626.45	707.61	L2(600)	218263.00	219265.10	1427.90
G3(400)	170924.00	171067.61	1237.18	L3(640)	506252.00	507468.06	1498.32
G4(480)	261993.00	262166.98	1480.98	L4(720)	659440.00	661788.10	1561.05
G5(200)	114163.64	114264.27	251.94	L5(760)	251711.00	252270.80	2164.20
G6(280)	140430.09	140568.73	469.81	L6(800)	834861.00	836689.50	1948.69
G7(360)	179378.00	180124.94	654.80	L7(840)	263299.00	264877.60	2249.60
G8(440)	193689.00	193954.60	1067.26	L8(880)	1029060.00	1031879.01	2372.60
G9(255)	$^{+}$4723.95	4730.46	209.47	L9(960)	1243250.00	1250196.50	2589.60
G10(323)	6712.53	6724.12	437.66	L10(1040)	1479270.00	1481956.70	2687.10
				L11(1120)	1735980.00	1738516.50	2989.30
				L12(1200)	2013640.00	2016821.40	3534.50
Average					284390.45	285088.41	904.95
Avg Gap with 5 runs (%)					0.01	0.03	

$\text{Gap}(sa_{Best}) = (sa^{5}_{Best} - sa^{10}_{Best})/sa^{10}_{Best} \times 100$, $\text{Gap}(sa_{Avg}) = (sa^{5}_{Avg} - sa^{10}_{Avg})/sa^{10}_{Avg} \times 100$.
Underline means this value is better than the C_{Best} for 5 runs, $^{+}$ means this value is not the BKS (see Table 3).

outperform the recently published methods by obtaining many new best results when tested on the benchmark data sets. New results for the larger VRP instances, which have not been previously tested for the CCVRP, are also reported.

In addition to the original min-sum objective, this study also considers the min-max CCVRP where appropriate adaptations of our algorithm are introduced. Interesting results are reported for the first time on many test instances. It was empirically found that the gap between the min-sum and the min-max results is rather small, especially when the data points exhibit some geometric structures. However, the maximum arrival time tends to increase at a higher rate when the min-sum objective is considered compared to a relatively smaller increase in the sum of arrival times when the min-max objective is applied. From a managerial decision making viewpoint, the min-sum objective will be preferable if one aims to reduce the overall customer arrival times, while the min-max will be more appropriate in the context of meeting deadline especially for the last customer.

A hybridisation of exact method and powerful metaheuristic would be an interesting challenge that could be explored to solve both the min-sum and the min-max CCVRP. For an easy to read overview on heuristics hybridisation and other related implementation issues, see the recent

Table 12: Results for the min-max CCVRP using 10 runs.

#(n)	ma_{Best}	ma_{Avg}	T_{Avg} (s)	#(n)	ma_{Best}	ma_{Avg}	T_{Avg} (s)
C1(50)	90.20	90.25	9.66	G11(399)	<u>40.70</u>	40.82	548.45
C2(75)	62.96	63.04	12.08	G12(483)	<u>46.67</u>	47.97	604.39
C3(100)	85.55	85.69	35.64	G13(252)	25.10	25.32	95.72
C4(150)	65.00	65.18	58.40	G14(320)	<u>29.10</u>	29.38	154.87
C5(199)	56.42	56.77	95.67	G15(396)	31.14	31.69	470.26
C11(120)	118.96	119.04	45.81	G16(480)	<u>33.94</u>	34.79	840.49
C12(100)	63.88	63.93	31.98	G17(240)	22.92	23.06	183.57
F1(44)	147.66	148.38	5.08	G18(300)	27.24	28.67	202.43
F2(71)	49.94	50.03	7.83	G19(360)	<u>31.62</u>	32.31	382.49
F3(134)	152.66	153.85	15.10	G20(420)	<u>38.58</u>	39.27	437.61
G1(240)	523.65	524.33	223.42	L1(560)	1514.21	1516.62	1094.68
G2(320)	734.85	735.98	715.36	L2(600)	871.84	873.13	1498.81
G3(400)	993.68	994.31	1319.42	L3(640)	1770.17	1771.06	1546.23
G4(480)	1252.51	1253.14	1500.47	L4(720)	<u>2029.00</u>	2032.03	1654.80
G5(200)	1234.21	1234.30	247.02	L5(760)	<u>778.08</u>	779.34	2167.42
G6(280)	1119.48	1120.14	467.63	L6(800)	2304.52	2305.82	2283.17
G7(360)	1171.67	1172.76	568.95	L7(840)	<u>725.40</u>	726.88	2674.11
G8(440)	<u>1046.99</u>	1047.67	1089.42	L8(880)	2558.00	2559.61	2638.40
G9(255)	32.48	32.69	194.32	L9(960)	2805.49	2807.64	2768.92
G10(323)	36.49	36.84	422.71	L10(1040)	<u>3064.32</u>	3068.17	2968.15
				L11(1120)	<u>3323.15</u>	3326.15	3298.30
				L12(1200)	<u>3586.69</u>	3590.05	3681.08
Average					826.12	827.10	934.77
Avg Gap with 5 runs (%)					0.06	0.09	

Underline means this value is better than the ma_{Best} for 5 runs.

monograph by Salhi (2017). From a practical point of view, a multi-objective optimisation routing problem that combines the VRP, min-sum and min-max objectives could also be worth attempting. A related problem that focuses on the environmental aspect by considering the accumulated load (demand) of the vehicles, as studied by Zachariadis et al. (2015), would be a useful extension that deserves exploration.

Acknowledgements

We would like to thank the editor and the reviewers for their useful comments and suggestions that improved the presentation as well as the content of the paper. The first author would also like to thank the Ministry of Higher Education Malaysia for her PhD scholarship.

References

Applegate, D., Cook, W., Dash, S., Rohe, A., 2001. Solution of a Min-Max Vehicle Routing Problem. INFORMS Journal on Computing 14 (2), 132–143.

Bowerman, R., Hall, B., Calamai, P., 1995. A multi-objective optimization approach to urban

school bus routing: Formulation and solution method. Transportation Research Part A: Policy and Practice 29 (2), 107–123.

Campbell, A. M., Vandenbussche, D., Hermann, W., 2008. Routing for Relief Efforts. Transportation Science 42 (2), 127–145.

Christofides, N., Mingozzi, A., Toth, P., 1979. The vehicle routing problem. In: Christofides, N., Mingozzi, A., Toth, P., Sandi, C. (Eds.), Combined Optimization. Wiley, Chichester, pp. 315–338.

Cinar, D., Gakis, K., Pardalos, P. M., 2016. A 2-phase constructive algorithm for cumulative vehicle routing problems with limited duration. Expert Systems With Applications 56, 48–58.

CLHO, 2016. Data sets - Routing Data Sets - VRP, Centre for Logistics and Heuristic Optimisation, Kent Business School, University of Kent.
URL `http://www.kent.ac.uk/kbs/research/research-centres/clho/datasets.html`

Corberán, A., Fernández, E., Laguna, M., Martí, R., 2002. Heuristic solutions to the problem of routing school buses with multiple objectives. Journal of the Operational Research Society 53 (4), 427–435.

Elshaikh, A., Salhi, S., Nagy, G., 2015. The continuous p-centre problem: An investigation into variable neighbourhood search with memory. European Journal of Operational Research 241 (3), 606–621.

Fakcharoenphol, J., Harrelson, C., Rao, S., 2007. The k -Traveling Repairmen Problem. ACM Transactions on Algorithms 3 (4), 40:1–40:16.

Fisher, M., 1994. Optimal solution of vehicle routing problems using minimum K-trees. Operations Research 42 (4), 626–642.

Goel, A., Gruhn, V., 2008. A General Vehicle Routing Problem. European Journal of Operational Research 191 (3), 650–660.

Golden, B., Wasil, E., Kelly, J., Chao, I., 1998. The Impact of Metaheuristics on Solving the Vehicle Routing Problem: Algorithms, Problem Sets, and Computational Results. In: Crainic, T. G., Laporte, G. (Eds.), Fleet management and logistics. Springer US, Boston, MA, pp. 33–56.

Golden, B. L., Kovacs, A. A., Wasil, E. A., 2014. Vehicle Routing Applications in Disaster Relief. In: Toth, P., Vigo, D. (Eds.), Vehicle Routing: Problems, Methods, and Applications, 2nd Edition. SIAM, Philadelphia, Ch. 14, pp. 409–436.

Hartl, R., Hasle, G., Janssens, G., 2006. Special issue on Rich Vehicle Routing Problems, editorial. Central European Journal of Operations Research 14 (2), 103–104.

Hof, J., Schneider, M., Goeke, D., 2017. Solving the battery swap station location-routing problem with capacitated electric vehicles using an AVNS algorithm for vehicle-routing problems with intermediate stops. Transportation Research Part B 97, 102–112.

Kara, I., Kara, B. Y., Yetis, M. K., 2008. Cumulative vehicle routing problems. In: Caric, T., Gold, H. (Eds.), The vehicle routing problem. In-Teh, Croatia, Ch. 6, pp. 85–98.

Ke, L., Feng, Z., 2013. A two-phase metaheuristic for the cumulative capacitated vehicle routing problem. Computers & Operations Research 40 (2), 633–638.

Koç, Ç., Bektaş, T., Jabali, O., Laporte, G., 2016. The impact of depot location, fleet composition and routing on emissions in city logistics. Transportation Research Part B 84, 81–102.

Lahyani, R., Khemakhem, M., Semet, F., 2015. Rich vehicle routing problems : From a taxonomy to a definition. European Journal of Operational Research 241 (1), 1–14.

Li, F., Golden, B., Wasil, E., 2005. Very large-scale vehicle routing: new test problems, algorithms, and results. Computers & Operations Research 32 (5), 1165–1179.

Li, J., Pardalos, P. M., Sun, H., Pei, J., Zhang, Y., 2015. Iterated local search embedded adaptive neighborhood selection approach for the multi-depot vehicle routing problem with simultaneous deliveries and pickups. Expert Systems with Applications 42 (7), 3551–3561.

Lin, S., 1965. Computer Solutions of the Traveling Salesman Problem. Bell System Technical Journal 44 (10), 2245–2269.

López-Sánchez, A. D., Hernández-Díaz, A. G., Vigo, D., Caballero, R., Molina, J., 2014. A multi-start algorithm for a balanced real-world Open Vehicle Routing Problem. European Journal of Opeational Research 238, 104–113.

Luo, Z., Qin, H., Lim, A., 2014. Branch-and-price-and-cut for the multiple traveling repairman problem with distance constraints. European Journal of Operational Research 234 (1), 49–60.

Lysgaard, J., Wøhlk, S., 2014. A branch-and-cut-and-price algorithm for the cumulative capacitated vehicle routing problem. European Journal of Operational Research 236 (3), 800–810.

Martínez-Salazar, I., Angel-Bello, F., Alvarez, A., 2015. A customer-centric routing problem with multiple trips of a single vehicle. Journal of the Operational Research Society 66, 1312–1323.

Méndez-Díaz, I., Zabala, P., Lucena, A., 2008. A new formulation for the Traveling Deliveryman Problem. Discrete Applied Mathematics 156 (17), 3223–3237.

Mladenović, N., Hansen, P., 1997. Variable neighborhood search. Computers & Operations Research 24 (11), 1097–1100.

Ngueveu, S. U., Prins, C., Wolfler Calvo, R., 2010. An effective memetic algorithm for the cumulative capacitated vehicle routing problem. Computers & Operations Research 37 (11), 1877–1885.

Pacheco, J., Mart, R., 2006. Tabu search for a multi-objective routing problem. Journal of the Operational Research Society 57, 29–37.

Park, J., Kim, B.-I., 2010. The school bus routing problem: A review. European Journal of Operational Research 202 (2), 311–319.

Ribeiro, G. M., Laporte, G., 2012. An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. Computers & Operations Research 39 (3), 728–735.

Rivera, J. C., Afsar, H. M., Prins, C., 2014. Multistart Evolutionary Local Search for a Disaster Relief Problem State of the Art. In: Legrand, P., Corsini, M., Hao, J., Monmarché, N., Lutton, E., Schoenauer, M. (Eds.), Artificial Evolution. EA 2013. Lecture Notes in Computer Science, vol 8752. Springer, Cham, pp. 129–141.

Rivera, J. C., Afsar, H. M., Prins, C., 2015. A multistart iterated local search for the multitrip cumulative capacitated vehicle routing problem. Computational Optimization and Applications 61 (1), 159–187.

Rivera, J. C., Afsar, H. M., Prins, C., 2016. Mathematical formulations and exact algorithm for the multitrip cumulative capacitated single-vehicle routing problem. European Journal of Operational Research 249 (1), 93–104.

Ropke, S., Pisinger, D., 2006. An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. Transportation Science 40 (4), 455–472.

Salehipour, A., Sörensen, K., Goos, P., Bräysy, O., 2008. An efficient GRASP+VND metaheuristic for the traveling repairman problem. Working Paper. University of Antwerp, Faculty of Applied Economics.

Salhi, S., 2017. Heuristic Search: The Emerging Science in Problem Solving. Palgrave, London.

Salhi, S., Sari, M., 1997. A multi-level composite heuristic for the multi-depot vehicle fleet mix problem. European Journal of Operational Research 103, 95–112.

Silva, M. M., Subramanian, A., Vidal, T., Ochi, L. S., 2012. A simple and effective metaheuristic for the Minimum Latency Problem. European Journal of Operational Research 221 (3), 513–520.

Stenger, A., Vigo, D., Enz, S., Schwind, M., 2013. An Adaptive Variable Neighborhood Search Algorithm for a Vehicle Routing Problem Arising in Small Package Shipping. Transportation Science 47 (1), 64–80.

Sze, J. F., Salhi, S., Wassan, N., 2016. A hybridisation of adaptive variable neighbourhood search and large neighbourhood search: Application to the vehicle routing problem. Expert Systems with Applications 65, 383–397.

Van Hentenryck, P., Bent, R., Coffrin, C., 2010. Strategic Planning for Disaster Recovery with Stochastic Last Mile Distribution. In: International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming. Springer Berlin Heidelberg, pp. 318–333.

Vidal, T., Crainic, T. G., Gendreau, M., Prins, C., 2014. A unified solution framework for multi-attribute vehicle routing problems. European Journal of Operational Research 234 (3), 658–673.

Wei, L., Zhang, Z., Lim, A., 2014. An Adaptive Variable Neighborhood Search for a Heterogeneous Fleet Vehicle Routing Problem with Three-Dimensional Loading Constraints. Computational Intelligence Magazine, IEEE 9 (4), 18–30.

Zachariadis, E. E., Tarantilis, C. D., Kiranoudis, C. T., 2015. The load-dependent vehicle routing problem and its pick-up and delivery extension. Transportation Research Part B 71, 158–181.