A Voronoi neighborhood-based search heuristic for distance/capacity constrained very large vehicle routing problems

Zhixiang Fang¹, Wei Tu¹, Qingquan Li¹ b, Shih-Lung Shaw¹ c, Shunqing Chen d & Bi Yu Chen b

¹ State Key Laboratory of Information Engineering in Surveying, Mapping, and Remote Sensing, Wuhan University, Wuhan, PR, China
² Engineering Research Center for Spatio-Temporal Data Smart Acquisition and Application, Ministry of Education of China, Wuhan, China
³ Department of Geography, University of Tennessee, Knoxville, TN, USA
⁴ CEO, Guangzhou Augur Intelligent Technology Co. Ltd., Guangzhou, PR China

Published online: 16 Aug 2012.

To cite this article: Zhixiang Fang, Wei Tu, Qingquan Li, Shih-Lung Shaw, Shunqing Chen & Bi Yu Chen (2013): A Voronoi neighborhood-based search heuristic for distance/capacity constrained very large vehicle routing problems, International Journal of Geographical Information Science, 27:4, 741-764

To link to this article: http://dx.doi.org/10.1080/13658816.2012.707319

PLEASE SCROLL DOWN FOR ARTICLE

Full terms and conditions of use: http://www.tandfonline.com/page/terms-and-conditions

This article may be used for research, teaching, and private study purposes. Any substantial or systematic reproduction, redistribution, reselling, loan, sub-licensing, systematic supply, or distribution in any form to anyone is expressly forbidden.
The publisher does not give any warranty express or implied or make any representation that the contents will be complete or accurate or up to date. The accuracy of any instructions, formulae, and drug doses should be independently verified with primary sources. The publisher shall not be liable for any loss, actions, claims, proceedings, demand, or costs or damages whatsoever or howsoever caused arising directly or indirectly in connection with or arising out of the use of this material.
A Voronoi neighborhood-based search heuristic for distance/capacity constrained very large vehicle routing problems

Zhixiang Fang\textsuperscript{a,*}, Wei Tu\textsuperscript{a,*}, Qingquan Li\textsuperscript{a,b,*}, Shih-Lung Shaw\textsuperscript{a,c}, Shunqing Chen\textsuperscript{d} and Bi Yu Chen\textsuperscript{b}

\textsuperscript{a}State Key Laboratory of Information Engineering in Surveying, Mapping, and Remote Sensing, Wuhan University, Wuhan, PR China; \textsuperscript{b}Engineering Research Center for Spatio-Temporal Data Smart Acquisition and Application, Ministry of Education of China, Wuhan, China; \textsuperscript{c}Department of Geography, University of Tennessee, Knoxville, TN, USA; \textsuperscript{d}CEO, Guangzhou Augur Intelligent Technology Co. Ltd., Guangzhou, PR China

(Received 29 March 2012; final version received 24 June 2012)

Local search heuristics for very large-scale vehicle routing problems (VRPs) have made remarkable advances in recent years. However, few local search heuristics have focused on the use of the spatial neighborhood in Voronoi diagrams to improve local searches. Based on the concept of a k-ring shaped Voronoi neighbor, we propose a Voronoi spatial neighborhood-based search heuristic and algorithm to solve very large-scale VRPs. In this algorithm, k-ring Voronoi neighbors of a customer are limited to building and updating local routings, and rearranging local routings with improper links. This algorithm was evaluated using four sets of benchmark tests for 200–8683 customers. Solutions were compared with specific examples in the literature, such as the one-depot VRP. This algorithm produced better solutions than some of the best-known benchmark VRP solutions and requires less computational time. The algorithm outperformed previous methods used to solve very large-scale, real-world distance constrained capacitated VRP.

Keywords: graph theory; large-scale optimization; simulated annealing; vehicle routing problem; heuristics

1. Introduction

The vehicle routing problem (VRP) is a frequently encountered optimization problem in many applications such as school bus routing, package delivery, and meter readings. Laporte (2009) defines the VRP as ‘the problem of designing least-cost delivery routes from a depot to a set of geographical scattered customers, subject to side constraints’ (p. 408). Although the VRP is one of the most extensively studied problems in optimization, use of spatial analysis and GIS to facilitate the search of spatial neighbors to improve the efficiency of VRP algorithms is somewhat limited. For example, is it possible to apply the Voronoi diagram concept to improve the computational speed and the solution quality of large-scale VRPs? This is a research topic that has the potential of making GIS more relevant to the spatial optimization problems. According to Golden \textit{et al.} (1998) and Zachariadis and Kiranoudis (2010), large-scale instances in optimization problems

\*Corresponding author. Email: zxfang@whu.edu.cn; tuweiwhu@gmail.com; qqli@whu.edu.cn

© 2013 Taylor & Francis
include 200–500 costumers, whereas Li et al. (2005) and Zachariadis and Kiranoudis (2010) suggest very large-scale instances as having more than 500 customers.

Generally, VRP supports the solving of path cover problems (Li et al. 2009), route optimization (Mooney and Winstanley 2006, Choi et al. 2009, Chen et al. 2012a, 2012b) and allocation (Hu et al. 2012) problems, and spatial decision-making for large-scale vehicles or persons in GIS. Numerous exact algorithms and heuristic algorithms (Laporte 2009) have been developed to solve the VRP over the past 50 years. Most exact algorithms are time-consuming and have a limited capability to address large-scale VRPs (Yücenur and Demirel 2011), particularly real-life cases such as transport delivery routing, school bus routing, urban trash collection, merchandise transport routing, hazardous product distribution, and tour planning for mobile health care facilities (see Jozefowiez et al. 2008).

To produce high-quality solutions of (large-scale) VRPs with less computational effort, researchers have developed numerous algorithms for classical heuristics and metaheuristics (Laporte 2009). A main difference between the classical heuristics and metaheuristics is the improved mechanisms that allow the objective function to deteriorate from an iteration to the next. Laporte (2009) classified metaheuristics into local searches, population searches, and learning mechanisms. Local searches use diverse neighborhood search strategies to improve solutions, while population searches and learning mechanisms use artificial intelligence approaches (i.e., genetic algorithm or ant colony optimization algorithm) to find better solutions. Local searches usually rely more on spatial knowledge of neighbors than population searches and learning mechanisms. Many effective local search strategies and algorithms have been introduced, including tabu search (Glover 1997, Wassan et al. 2008, Bolduc et al. 2010, Brandão 2011), simulated annealing (SA) (Kirkpatrick et al. 1983, Hiquebran 1993), deterministic annealing (Dueck and Scheuer 1990, Kaku et al. 2003), variable neighborhood search (Mladenović and Hansen 1997, Hansen and Mladenović 2001, Kytöjoki et al. 2007, Fleszar et al. 2009, Imran et al. 2009), very large neighborhood search (Ergun et al. 2006, Lee et al. 2008), adaptive large neighborhood search (Ropke et al. 2006b, Milthers 2009), and so on. In addition, several very large-scale neighborhood search techniques were reviewed by Ahuja et al. (2002). To demonstrate the usefulness of Voronoi diagrams for solving VRPs, this article will concentrate on the improvement of local searches.

The size of neighborhoods being searched can affect the quality and speed of the local search algorithms. However, very few studies of local search algorithms have aimed at reducing the neighborhood size in VRP algorithms based on the concept of Voronoi neighborhoods (Fortune 1987). Local searches are frequently used to find good approximate solutions of VRPs within a reasonable time. Local search algorithms move from solution to solution within a neighborhood space by applying local changes and end when a solution, which is deemed to be optimal, has been found or a time limit has elapsed. There is a basic rule that a larger neighborhood space makes it possible for local search algorithms to find better quality, local optimal solutions (Pisinger and Ropke 2010), but it needs more time to search the neighborhood at each iteration of the VRP algorithms. For example, Milthers (2009) used Voronoi diagrams to divide the VRP with time windows (VRPTW) into subproblems, and extended the adaptive large neighborhood search framework of Ropke and Pisinger (2006a) to solve VRPTW. Milthers’ work showed the advantages of calculating small-scale subproblems of the VRPTW problem using Voronoi diagrams to guide the neighbor search during the process of removal and reinsertion. In contrast to Milthers (2009) in implementation, Zachhariaidis and Kiranoudis (2010) introduced the static move descriptor (SMD) data structures to reduce the complexity of neighborhood evaluation. Different from the above studies, this article presents a Voronoi spatial neighborhood-based
search heuristic for solving a variant of distance constrained capacitated VRPs (DCVRPs) without a dividing process. This variant has a hierarchical objective; it first minimizes fleet size and then the distance. This heuristic restricts the local search of DCVRP algorithms within a neighborhood space consisting of Voronoi neighbors. These neighbors are selected on the basis of a proposed concept of k-ring shaped Voronoi neighbors. Based on this heuristic, a Voronoi spatial neighborhood-based search heuristic algorithm (VSHA) was proposed to solve large-scale (as many as 8683 customers) DCVRPs.

The contributions of this work can be summarized as follows: (1) a Voronoi-based cluster-first route-second approach was introduced to generate the initial DCVRP solutions; (2) a concept of k-ring shaped Voronoi neighbors was introduced to reduce the size of the neighborhood used for a local search; (3) a Voronoi-based rearrangement strategy was introduced to improve local routes with improper links; and (4) the effectiveness of this algorithm was tested using published data sets and our large-scale DCVRP data sets. The algorithm produced excellent results quickly and improved several well-known solutions published in literature.

The article is organized as follows: Section 2 provides a brief literature review of neighborhood reduction strategies, neighborhood structures, and evolutionary algorithms for VRPs. Section 3 introduces the concept of a k-ring shaped Voronoi neighbor. Section 4 gives a detailed description of the proposed algorithm. Section 5 discusses the computational results of our algorithm. Finally, Section 6 draws the conclusions and discusses the future research directions.

2. Related work

This section gives a brief literature review of neighborhood reduction strategies, neighborhood structures, and several evolutionary algorithms for VRPs.

2.1. Neighborhood reduction strategies

Neighborhood search is a critical function in local search algorithms to define their search space. Researchers have introduced many strategies to reduce the neighborhood search space (Zachariadis and Kiranoudis 2010) and to explore neighborhood space efficiently, in order to accelerate local search algorithms and find high-quality solutions.

The first strategy is the candidate list strategy for narrowing the examination of neighborhood elements. A candidate list attempts to keep a limited number of the most likely used neighbors in each local search step. For example, Glover and Laguna (1997) introduced some classes of candidate list strategies that select several best moves encountered in a single iteration. Coy et al. (1998) used a fixed-length neighbor list strategy. Lourenço et al. (2010) adopted a fixed-radius, nearest-neighbor search restricted to candidate lists with a fixed number (i.e. 40) of nearest neighbors of each customer (i.e. city), and a ‘Don’t-look bids’ strategy (Bentley 1992, Martin and Otto 1996, Nagata and Bräysy 2008). Devarenne et al. (2008) used an adaptive candidate list strategy based on two candidate lists in the intensification and diversification phases of local search, respectively. Resende and Ribeiro (2010) created a restricted candidate list (RCL) strategy formed by the best elements with the smallest incremental costs. The RCL is limited either by the number of elements (cardinality based) or by their quality (value based).

The second strategy is the granular search, which ‘discards “long” (high-cost) arcs due to the small probability of these being part of high-quality solutions and concentrates on promising “short” arcs, being those arcs whose cost does not exceed a granularity
The granular search introduced by Toth and Vigo (2003) drastically reduces the size of the restricted neighborhoods and prevents the involvement of elements that are unlikely to belong to good feasible (optimal) solutions. The granular search may save time at the expense of the quality of the solution. It can be viewed as an efficient implementation of candidate list strategies. For example, Li et al. (2005) developed a record-to-record travel algorithm that uses a variable neighborhood list similar to the Toth and Vigo's (2003) granular search mechanism. By extending the idea of a granular search, Prins et al. (2007) explored an unused and non-tabu edge or linking between two customers in a reduced neighborhood, which only contains all feasible solutions in a single iteration. Branchini et al. (2009) proposed an adaptive granular local search heuristic to adjust the size of the search space for dynamic VRPs.

The third strategy is the aggregation of demands. This approach can be viewed as an approach to arc routing (Oppen and Løkketangen 2004). The aggregated demands of close customers reduce the size of the VRP, thus improving the chances of finding good solutions in a reasonable time. Hjertenes (2002) aggregated the demands of cities in travelling salesman problems by using structural information about the city locations. Oppen and Løkketangen (2004, 2006) aggregated the customers to reduce the size of the stringed VRP. Their results performed well in solving VRPs, especially for large instances with many customers per road segment. The aggregation strategy influences the solution quality to a large extent as different aggregation approaches generate different combinations of demand nodes in VRPs.

A ‘cluster-first route-second’ heuristic for the VRP is similar to the third strategy. This heuristic clusters all demand nodes into several feasible groups and then proposes individual routes for each group. For example, Mole et al. (1983) analyzed the ‘route-first cluster-second’ heuristic and determined its efficiency when searching the VRP solution space. Hiquebran et al. (1993) presented an SA algorithm with a cluster-first route-second heuristic for the VRP. Prutsakul (1998) expanded this heuristic by clustering retailers according to their replenishment periods and routing clustered retailers using a nearest-neighbor algorithm.

The candidate list and the granular search strategies can reduce the size of search space for VRP. However, they still include elements that are unlikely to be good feasible (optimal) solutions due to their simple approach of using distance to reduce the search space. The aggregation of demands and the cluster-first route-second heuristic methods solve large-scale VRPs by replacing the solution of a large-scale VRP with solutions of several small VRPs, whereas the aggregation approach uses arbitrarily selected initial solution that could significantly influence the solution quality. This article extends the candidate list and the granular search strategies to solve large-scale DCVRPs by limiting the search space based on a concept of k-ring shaped Voronoi neighbor, and expands the cluster-first route-second heuristic to generate initial VRP solution with the constraint of Voronoi neighbor.

### 2.2. Neighborhood structure

Neighborhood structure (i.e. the manner in which the neighborhood is defined) in local searches and metaheuristics is crucial to the efficiency of VRP algorithms and the quality of their solutions. A systematic change of neighborhood structures provides solution spaces with improved routings for VRP algorithms. Different neighborhood structures are conducted in either a deterministic or a stochastic manner. An effective heuristic for VRPs needs a rational neighborhood structure. Numerous neighborhood structures have been introduced in the literature including some well-known VRP inter-route neighborhood
structures [interchange (swap), relocate, exchange, cross, 2-opt*, b-cyclic, k-transfer (ejection chains), GENI (Gendreau et al. 1992), etc.] and intra-route neighborhood structures (2-opt and 3-opt moves, Or-opt, etc.). These inter-route and intra-route neighborhood structures offer several improved solution mechanisms for rearranging the local neighborhood routings constrained by the objective functions (i.e. time, cost, or distance). For example, Van Breedam (1994) classified the move and cross-exchange neighborhood structures for vehicle routing. Kytöjoki et al. (2007) used seven neighborhood structures (2-opt, Or-opt, 3-opt, exchange, relocate, 2-opt*, and cross-exchange) in their guided variable neighborhood search method that was capable of escaping local minima. Hemmelmayr et al. (2009) applied the 3-opt neighborhood structure in their proposed variable neighborhood search heuristic for the periodic routing problem. Zachariadis and Kiranoudis (2010) reviewed three common neighborhood structures (1–0 exchange, 1–1 exchange, and 2-opt). Groër et al. (2010) implemented six neighborhood structures (1 point move, 2-opt move, or-opt move, 3-opt move, 3 points move, and cross-exchange move) in their library of VRP heuristics (VRPH). This article implemented six neighborhood structures (1–0 exchange, 1–1 exchange, 3 points move, 2-opt and 3-opt moves, and Or-opt) (Groër et al. 2010, Zachariadis and Kiranoudis 2010) using Voronoi diagrams to optimize the DCVRP.

2.3. Evolutionary algorithms for VRP

Besides the local search algorithms, Laporte (2007, 2009) also concluded two types of important metaheuristics: population search and learning mechanisms. Both of these can be viewed as evolutionary algorithms. Some evolutionary algorithms (genetic algorithm, ant colony optimization algorithm, etc.) were implemented to solve the VRPs. The well-known genetic algorithms (Holland 1975) use the concepts of crossover, mutation, and selection to improve the solution. Almost all known genetic algorithms applied to VRPs use local searches. Some recent implementations are provided in Baker and Ayechew (2003), Prins (2004), Marinakis and Marinaki (2010), and Ursani et al. (2011). Ant colony optimization algorithms (Blum 2005) use a pheromone update mechanism to guide the shortest paths followed by ants. This algorithm increases the importance of exploiting information collected by previous ants with respect to exploration of the search space (Dorigo and Stützle 2010). Some recent implementations for VRPs have been provided by Santos et al. (2010), Balseiro et al. (2011), and Yu and Yang (2011). In addition, evolutionary algorithms including some promising approaches, such as particle swarm optimization (Marinaki and Marinaki 2010) and differential evolution (Cao and Lai 2009), also were introduced to solve large-scale VRPs. Efficient local search strategies can also be integrated in these evolutionary algorithms. Moreover, there are some optimization algorithms in GIS for VRP-related problems. For example, several spatial decision support systems or applications in GIS (Tarantilis et al. 2004, Keenan 2008, Mendoza et al. 2009, Lei and Church 2010, Santos et al. 2011) were introduced for vehicle routings. Li and Yeh (2005) integrated genetic algorithms and GIS for optimal location search. Liu et al. (2008) and Liu et al. (2012a) used ant colony optimization method to optimize land use in GIS, and Liu et al. (2012b) used swarm intelligence to optimize ecological areas. Most of their works could be extended as VRP-derived GIS applications. Although evolutionary algorithms have potential of improving vehicle routings solutions, this article will concentrate only on the improvement of local searches in order to demonstrate the usefulness of Voronoi diagrams for solving VRPs. Therefore, this study focuses on a Voronoi spatial neighborhood-based search heuristic, which is integrated into an SA algorithm (Kirkpatrick et al. 1983, Osman 1993) in order to test its performance.
3. The concept of the k-ring shaped Voronoi neighbor

This section introduces the concept of k-ring shaped Voronoi neighbor used in the following algorithm for modeling the neighborhood space in each local search step. A Voronoi diagram (Aurenhammer 1991, Bakolas and Tsiotras 2010) is a typical form of metric space decomposition according to a nearest-neighbor rule that each point is associated with the region of the plane close to it. A Voronoi cell for a site \( p \) is the set of points in the plane that are closer to this site \( p \) than to any other site in the Voronoi diagram. A Voronoi region is a convex polytope acting as the boundary of this set of points in the plane. Detailed definitions of Voronoi diagrams can be found in Aurenhammer (1991) and Okabe et al. (2000). By extending these basic concepts of Voronoi diagrams, this section introduces the following four definitions:

**Definition 1:** A site \( p \) has its Voronoi regions \( VR(p) \) and another site \( q \) has its Voronoi regions \( VR(q) \). If two Voronoi regions \( VR(p) \) and \( VR(q) \) share a Voronoi edge, then sites \( p \) and \( q \) are Voronoi neighbors to each other. The two sites are a Voronoi neighbor pair.

**Definition 2:** A Voronoi distance (Duczmal et al. 2011) between two sites \( p \) and \( q \) is defined as the number of Voronoi region boundaries that must be crossed in order to establish a path between \( p \) and \( q \). For example, the Voronoi distance between \( p \) and \( q \) in Figure 1 is 1.

**Definition 3:** A Voronoi neighbor set \( VN \) consists of all Voronoi neighbors of a site \( p \); the set \( VN \) is composed of the 1-ring shaped Voronoi neighbors denoted by \( VN(p,1) \). For example, the Voronoi neighbors in the Voronoi regions, which have same legend as the Voronoi region of \( q \) in Figure 1, are 1-ring shaped Voronoi neighbors of \( p \).

![Figure 1. k-Ring shaped Voronoi neighbors.](image_url)
**Definition 4:** The k-ring shaped Voronoi neighbors of a site \( p \) are the Voronoi neighbors of \( VN(p,k - 1) \) and do not include these Voronoi neighbors \( VN(p,k - 2) \). The 1-ring, 2-ring, 3-ring, 4-ring, and 5-ring Voronoi neighbors are shown in Figure 1.

The definition of k-ring shaped Voronoi neighbors helps to make it easy to identify the (spatial) neighbors of each customer during the local search of the proposed algorithm in Section 4. This algorithm will only maintain the candidate list consisting of \( VN(p,k) \), \( VN(p,k - 1) \), \( VN(p,k - 2) \), \ldots, \( VN(p,1) \) Voronoi neighbors to reduce the computational effort in each neighborhood search.

**4. The proposed algorithm**

The proposed VSHA belongs to the class of SA algorithms in conjunction with local search methods. ‘Simulated annealing is a generic probabilistic metaheuristic for the global optimization problem of locating a good approximation to the global optimum of a given function in a large search space’ (http://en.wikipedia.org/wiki/Simulated_annealing). Additional information of SA can be found in literature (Kirkpatrick et al. 1983, Hiquebran 1993). The proposed VSHA is an extension of SA by using the strategies and knowledge derived from Voronoi diagrams. This algorithm limits the local search among k-ring shaped Voronoi neighbors and implements a systematic change of neighborhoods through the rearrangement of Voronoi spatial neighbors during routings of the solution. An overview of the proposed algorithm is introduced in Section 4.1 and the main steps are explained in Section 4.2.

**4.1. An overview of the proposed algorithm**

Let \( G = (V, E) \) denote a complete graph, where \( V = \{v_0, v_1, \ldots, v_n\} \) is a customer set and \( E = \{(v_i, v_j) \mid v_i, v_j \in V, i \neq j \text{ and } i, j \leq n\} \) is the arc set; \( v_i \) is the \( i \)-th customer in this set; \( v_0 \) is the depot in the DCVRP; \( n \) is the number of customers; \( m \) is the size of a fleet of identical vehicles; and \( Q \) is the capacity of a vehicle. Each customer \( i \) has a non-negative demand \( q_i \). \( C_{i,j} \) is the cost from the customer \( i \) to \( j \), which is associated with the arc \( (v_i, v_j) \). The DCVRP is an optimization problem to determine which customers are served by which vehicle and which route the vehicle follows to serve those customers, while minimizing the operational costs (i.e., travel distance) of the fleet. Each vehicle does not exceed the capacity \( Q \), and each customer can only be served by one vehicle. \( S \) is a solution of DCVRP that consists of \( m \) routings. \( S = \{r_i \mid 1 \leq i \leq m\}, r_i = \{0, v_j, \ldots, v_j, 0\}, v_j \in V. \) Each route departs from the depot and returns to the depot. \( V(s) \) represents a Voronoi diagram derived from the set \( V \); \( l_{ij} \) is the distance between customers \( i \) and \( j \); \( Q(r_i) \) represents the current load of the vehicle \( r_i \); \( Cluster(i) \) is the cluster that includes customer \( i \); \( \text{max}_{\text{iter}} \) is the maximum iteration number in the SA algorithm; \( cool_{\text{ratio}} \) is the cooling ratio of this SA algorithm; \( k \) is the maximum ring of the Voronoi neighbors for a local search; \( t \) is the current temperature in the SA algorithm; and \( d \) and \( \alpha \) are, respectively, the Voronoi distance and angle thresholds used to control the local search. This algorithm is proposed to solve a variant of the DCVRP with a hierarchical objective where it minimizes the size of the fleet, then minimizes the distance.

Figure 2 provides an overview of the VSHA. The VSHA modified the initial and improvement phases of the SA algorithm. In the initialization step, a Voronoi diagram \( V(S) \) that tessellates the space of the corresponding customer \( V \) is initially created using the sweepline algorithm developed by Fortune (1987) with a computational complexity of
An initial solution is then generated according to a cluster-first route-second heuristic constrained by k-ring Voronoi neighbors. The second step of this algorithm improves the initial solution using two important procedures: reducing the routings and the distance by using an SA algorithm. The routing reducing procedure aims to reduce the number of vehicles in the initial solution until it is near or equal to the predefined number of vehicles in the DCVRP or the iteration is greater than the parameter max_iter. The procedure for reducing the length of the solution aims to improve the quality of the solution by rearranging routings with improper links. An improper link has two customers with a Voronoi distance of d that is greater than k, whereas the angle between the direction of the link and the direction of the ‘depot and the link’s customers is greater than α. In addition, the number of initial routings should be reduced as much as possible. The minimum initial vehicle number m of the DCVRP can be defined according to Equation (1) or can be predefined by users:

\[ m = \frac{\sum_{i=1}^{n} Q(v_i)}{Q} \]  

where \( Q(v_i) \) is the demand of the customer \( i \).
4.2. Explanation of the main steps

4.2.1. Creating the initial routings (step a)

Figure 3 provides the pseudocode used to create the initial solution. It includes three steps: clustering customers in a Voronoi diagram, generating an inner route for each cluster, and linking the inner routes as initial vehicle routes.

To cluster customers and generate an inner route for each cluster, this study built up a sorted heap ($SH$) containing all neighbor pairs of a Voronoi diagram. This heap is sorted using a heapsort algorithm (Paulik 1990) based on a criterion of the distance between two adjacent customers in the neighbor node pair. Each customer is initially allocated to a cluster based on the customer’s initial demand. The first neighbor pair $vp$ with the smallest distance is then popped. The two customers of $vp$ are merged into a cluster only if they meet the condition:

$$Q[\text{Cluster (} vp . n_i \text{)}] + Q[\text{Cluster (} vp . n_j \text{)}] \leq \frac{Q}{3} \quad (2)$$

---

**Creat_initial_solution** ($G, NS, V(S), k$)

**Sorted Heap** $SH, RH$

**Customer pair** $vp(n_i,n_j)$

**Capacity of fleet** $Q$

**Solution** $S^*$

**Route** $r, tmpr1, tmpr2$

Put all Voronoi neighbor pairs into $SH$

While ($SH$ is not empty)

- **Cluster customers of $V$ in $G$**
  
  if $Q(\text{Cluster (} vp . n_i \text{)}) + Q(\text{Cluster (} vp . n_j \text{)}) \leq Q/3$
  
  Merge the $\text{Cluster (} vp . n_i \text{)}$ and $\text{Cluster (} vp . n_j \text{)}$

  remove $vp$ from $SH$

- **Link customers $vp.n_i$ and $vp.n_j$ as route segments of $r$**

  Link the customers $vp.n_i$ and $vp.n_j$ to customers in $r$ using a strict constraint of the least incremental length of the current route

---

**Link inner routes as an initial route**

Put all inner routes into $RH$, and clear $r$

While ($RH$ is not empty)

- $tmpr1$ = the farthest inner route from the depot in $SH$

  $tmpr2$ = the nearest inner route of $tmpr1$ in $V(S)$

  if $Q(tmpr1) + Q(tmpr2) \leq Q$

    Link the begin and end customers of $tmpr1$ to $tmpr2$ using a strict constraint of the least incremental length

  else

    Put $tmpr1$ into $S^*$

    end if

remove $tmpr1$ from $SH$

return Local search($S^*, NS, V(S), k$)

---

Figure 3. Pseudocode used to create initial solution.
The capacity of this merged cluster equals the sum of the demands of the two clusters. The other neighbor pairs in $SH$ are then popped one by one to update the clusters. As a result, customers of the same color in Figure 4a are in the same cluster. At the same step, an inner route is generated for each cluster by linking the nodes of the cluster according to the sequence in which they are popped from the heap for all neighbor pairs. A link can be inserted in an inner route using a strict constraint of the least incremental length of the current route. The links connecting customers of the same color are the inner routes in Figure 4a.

The next step is to link the inner routes as an initial solution. The inner routes are linked in order, from the farthest to the nearest to the depot. The farthest route can be linked to its neighbor inner route if its capacity does not exceed the vehicle capacity $Q$. The start and end points of the two inner routes are linked according to a strict constraint on the least incremental length of the current route. This step can be repeated to link other inner routes and candidate vehicle routes. It is terminated by the condition that no inner routes can be combined using other routes. Figure 4b illustrates the linked routes.

4.2.2. Local search (step b)

The local search step is to optimize every route using the six improvement operators individually, that is, 1–0 exchange, 1–1 exchange, 2-opt, 3-opt, Or-opt, and 3 points move. All exchanges or moves are limited within maximum k-ring Voronoi neighbors. For example, the node $b$ in the routing $r1$ is relocated to routing $r2$ in Figure 5a (1–0 exchange).
Nodes $b$ and $j$ are swapped between routings $r_1$ and $r_2$ in Figure 5b (1–1 exchange). Routing $r_1$ is broken from nodes $b$, $i$, and $k$, which reorganizes these links in Figure 5c (3-opt). Nodes $b$ and $c$ in routing $r_1$ are moved to routing $r_2$ before node $j$ in Figure 5d (Or-opt). The remaining links from node $b$ in routing $r_1$ are swapped with the remaining links from node $j$ in routing $r_2$ (Figure 5e) (2-opt move). Nodes $b$ and $c$ in routing $r_1$ are moved to routing $r_2$ before node $j$ in Figure 5f (3 points move). Figure 4c shows the optimized routes by the local search. In each local search, the execution sequence of these six improvement operators follows the order of 1–0 exchange, 1–1 exchange, 2-opt, Or-opt, 3 points move, and 3-opt. For each improvement operator, we first randomly select node $b$ among all demand nodes. Nodes $j$ and $k$ are chosen within the maximum k-ring Voronoi neighbors of the node $b$; then these two or three nodes are used to change their routings according to the operators illustrated in Figure 5. This process is repeated $N$ times in each local search, where $N$ is the number of demand nodes. Each demand node therefore is processed with all improvement operators and each route is optimized by the 3-opt operator with $R$ times, where $R$ is the number of routes in the current solution. In order to be comparable with the six improvement operators reported in Groër et al. (2010), Zachariadis and Kiranoudis (2010), we implement the same six operators using the concept of k-ring shaped Voronoi neighbor to reduce neighborhood space in local search.

### 4.2.3. Acceptance condition (step c)

If the new solution $S'$ (e.g., $tmpS$ in Figure 2) is an improved version of the current solution $S^*$ (i.e., $f(S') - f(S^*) < 0$), this new solution $S'$ is accepted as the current solution $S^*$. When the new solution $S'$ is not better than the current solution $S^*$, $S'$ will be accepted as the current solution if both solutions satisfy the following condition:

$$e^{-\frac{f(S') - f(S^*)}{t}} < r$$

where $f(S')$ represents the total length of all routings in the solution $S'$, $t$ is the current temperature in the SA algorithm, and $r$ is a randomly generated number that is in the region 0–1. The temperature $t$ is used to control the convergence of the solution in an SA.

### 4.2.4. Inserting customers on the least loaded routing into nearby routings (step d)

Figure 6 gives the pseudocode used to insert customers into nearby routings. Neighbor routings of the least loaded routing are found based on the rule that there is at least one customer on neighbor routings, which is within the k-ring Voronoi neighbors of any customer in the least loaded routing. All customers of the least loaded routing will be inserted into its neighbor routings in a sequential order, from the first customer to the last customer of the least loaded routing. Each customer of the current least loaded routing is inserted into one of the identified neighbor routings based on a criterion of least incremental length. After each insertion, this customer is removed from the current least loaded routing. By repeating this step, all customers are inserted into these neighbor routings.

After these insertions, the capacity of $r'$ may be overloaded. The next step moves some customers from overloaded routings to neighbor routings that meet the requirement that each routing’s capacity should not exceed the vehicle capacity $Q$. The moving operation constrained within k-ring Voronoi neighbors is conducted using the 1–0 exchange or 1–1 exchange operators shown in Figure 6. A customer can be moved to a neighbor
routing if the neighbor routing has sufficient capacity to meet the demand of the customer and the least incremental length produced by the exchange operator.

4.2.5. Rearranging routings with improper links (step e)

Figure 7 gives the pseudocode of rearranging routings with improper links in the current solution, and Figure 8 illustrates the process of arranging routes with improper links. This procedure scans all the links in the current routing solution to find any improper links (e.g., the green link in Figure 8a) based on the constraints of user-predefined maximum values of \(d\) and \(\alpha\). The customers on improper links, the customers whose Voronoi regions are then

---

```plaintext
Insert_customers_into_nearby_routings (G, r, V(S), k)
Capacity of fleet \( \tilde{Q} \)
Customer set \( V \)
Route set \( S \)

Insert all customers of the least load routing \( r \) into \( V \)
Find the nearby routings \( S \) to all customers of \( V \) within k-ring
Voronoi neighbors of \( V(S) \)
While ( \( V \) is not empty)
    Find the route \( r' \) in \( S \) with the least incremental length if
    inserting the first customer \( v' \) in \( V \) into it
    Insert \( v' \) to \( r' \)
    if \( Q(r') > \tilde{Q} \)
        Move some customers from the \( r' \) to nearby routings
        remove \( v' \) from \( V \)
```

Figure 6. Pseudocode to insert customers into nearby routings.

---

```plaintext
Rearrange_routings_with_improper_links (S*, \( \alpha \), \( d \), V(S), k)
Capacity of fleet \( \tilde{Q} \)
Customer set \( V \)
Route set \( S \)

--Find improper links and customers for arrangement
Scan \( S^* \) to find any improper links with constraints of \( \alpha \), \( d \)
Insert customers of these links into \( V \)
Insert customers of these links acrossing its Voronoi regions into \( V \)
Insert k-ring,(k-1)-king,...,1-ring Voronoi neighbors of each current
customer of \( V \) into \( V \)
Insert the routings including customers of \( V \) into \( S \)
Remove customers of \( V \) from \( S \)

--Rearranging local routings near to each improper link
While ( \( V \) is not empty)
    Find the route \( r' \) in \( S \) with the least incremental length if
    inserting the first customer \( v' \) in \( V \) into it
    if \( Q(r') + Q(v') \leq \tilde{Q} \)
        Insert \( v' \) to \( r' \)
        remove \( r' \) from \( V \)
```

Figure 7. Pseudocode for rearranging routings with improper links.
crossed by these improper links, and their k-ring Voronoi neighbors of the above two types of customer are inserted into a customer set \( V \) (e.g., the isolated customers in Figure 8b). The routings including these customers are inserted into a route set \( S \). The customers in \( V \) are removed from the routings in \( S \). Each customer in \( V \) is inserted into a routing in \( S \) based on the criterion of the least incremental length. The local routings are arranged after inserting all the customers of \( V \) into \( S \). These local routings and the unchanged routings of the current solution \( S \) thus form a new solution \( S' \) (see Figure 8c).

5. Computational results
Computational tests were carried out in order to assess the performance and determine the parameter setting of the proposed VSHA. We tested it on four large and very large-scale (real-world) VRP benchmarks. Section 5 also discusses the parameter settings. We provide our solutions and compare the VSHA with two other state-of-the-art algorithms.

5.1. Benchmark testing
There are four benchmark tests in this study. The first test problems include 20 of the Golden benchmarks with 200–483 customers (Golden et al. 1998). The main characteristic of these benchmarks is that their customers are regularly distributed in a space. The second set of benchmark tests are the 12 very large-scale VRPs from Li et al. (2005) with 560–1200 customers. Each problem exhibits geometric symmetry. The third set of benchmark tests are the eight very large-scale real-world examples from Zachariadis and Kiranoudis (2010) with 3000 customers. The fourth set of benchmark tests includes four logistical routing problems from a company in Guangzhou, China, with 4594–8683 customers. The last two instances that were tested share an irregular distribution of customers in space.

5.2. Two state-of-the-art algorithms for comparisons
Two state-of-the-art algorithms found in the literature were compared with the proposed VSHA. The first algorithm was a penalized SMD algorithm (PSMDA) developed by Zachariadis and Kiranoudis (2010). Results of the PSDMA are directly referred to in this article. The second algorithm was based on ejection and injecting strategies for VRPs (VRP_EJ), which is an algorithm released in an open source VRPH software package for
the VRP (http://www.coin-or.org/projects/VRPH.xml). The parameters of the algorithm were introduced by Groër et al. (2010). For example, the size of the neighbor list searched and inserted when running the local search operators was 30. The computational results of the VSHA and the VRP_EJ algorithm reported here were obtained by compiling this algorithm with the C++ programming language and running them on a personal computer with an Intel(R) Core(TM)2 Duo CPU T7250 @2.00 GHz, 2.00 GHz, and 2.00GB of RAM. Only one CPU was used in this study. The computational results of the PSMDA reported by Zachariadis and Kiranoudis (2010) were implemented in C# and executed on the single core of a T5500 processor (1.66 GHz). All computational time for the PSMDA were calculated with a coefficient \(0.899365 = \frac{992}{1103}\) (Dongarra’s factors for the two CPUs are derived from http://www.cpubenchmark.net. Dongarra’s factor is used to convert the CPU time of different machines that support the comparison of computing speeds among different algorithms) by considering an indicatively relative speed as established by Dongarra (2011). In addition, Morin (2002) tested the difference of computation time between C++ and C# and showed that the integer calculations are approximate twice as fast in C++ as C#. This is partially due to the fact that an algorithm written in C++ and compiled would run much faster than the same written in C# and runs as an interpreter.

5.3. Parameter testing

Three main parameters (i.e., the Voronoi distance \(d\), the angle threshold \(\alpha\), and the maximum iteration threshold \(\text{max\_iter}\)) of the VSHA were tested using a ZK1 very large-scale, real-world DCVRP (Zachariadis and Kiranoudis 2010) because this algorithm was also proposed to solve the real-world DCVRP. Table 1 lists the test results produced with different combinations of the parameters \(\alpha\) and \(d\). The length in this table is the total routing length of a solution. The total routing length of a solution was the lowest in Table 1 when \(d = 3\) and \(\alpha = 40\). However, its corresponding computational time (2392.3 s) was greater than in cases where \(d < 3\). The test result when \(d = 2\) and \(\alpha = 20\) was better than other cases where \(d < 3\). The computational time for case \(d = 2\) and \(\alpha = 20\) is 1445.8 s. Due to the importance of cost performance for large-scale, real-world DCVRPs, this study selected \(d = 2\) and \(\alpha = 20\) as the parameters for the VSHA when solving real-world DCVRPs.

Table 1. Results of parameter testing.

<table>
<thead>
<tr>
<th>Voronoi distance ((d))</th>
<th>Angle threshold ((\alpha))</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Length</td>
<td>13,641.90</td>
<td>13,593.78</td>
<td>13,632.74</td>
<td>13,598.35</td>
<td>13,598.59</td>
<td>13,613.07</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>835.0</td>
<td>853.9</td>
<td>852.5</td>
<td>846.1</td>
<td>860.2</td>
<td>849.6</td>
</tr>
<tr>
<td>2</td>
<td>Length</td>
<td>13,539.77</td>
<td>13,532.12</td>
<td>13,536.26</td>
<td>13,551.08</td>
<td>13,545.52</td>
<td>13,540.95</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>1462.7</td>
<td>1445.8</td>
<td>1442.0</td>
<td>1455.1</td>
<td>1446.8</td>
<td>1450.5</td>
</tr>
<tr>
<td>3</td>
<td>Length</td>
<td>13,561.60</td>
<td>13,540.65</td>
<td>13,544.74</td>
<td>13,526.48</td>
<td>13,538.83</td>
<td>13,542.46</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>2394.9</td>
<td>2388.1</td>
<td>2390.44</td>
<td>2392.3</td>
<td>2396.1</td>
<td>2392.4</td>
</tr>
<tr>
<td>4</td>
<td>Length</td>
<td>13,541.24</td>
<td>13,538.38</td>
<td>13,559.35</td>
<td>13,542.81</td>
<td>13,540.57</td>
<td>13,544.47</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>3819.2</td>
<td>4368.3</td>
<td>4512.9</td>
<td>4548.4</td>
<td>4080.5</td>
<td>4265.8</td>
</tr>
<tr>
<td>5</td>
<td>Length</td>
<td>13,558.91</td>
<td>13,542.63</td>
<td>13,558.60</td>
<td>13,531.54</td>
<td>13,566.39</td>
<td>13,551.61</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>5607.1</td>
<td>6047.4</td>
<td>6091.0</td>
<td>5988.7</td>
<td>6263.7</td>
<td>5999.6</td>
</tr>
<tr>
<td>Mean</td>
<td>Length</td>
<td>13,568.69</td>
<td>13,549.51</td>
<td>13,566.34</td>
<td>13,550.05</td>
<td>13,557.98</td>
<td>13,557.98</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>2823.8</td>
<td>3020.7</td>
<td>3057.8</td>
<td>3046.1</td>
<td>3009.4</td>
<td>3009.4</td>
</tr>
</tbody>
</table>
This study tested the total routing length in different iterations using the VSHA to solve ZK1. In this computation, the VSHA used the parameters \( d = 2 \) and \( \alpha = 20 \). The results in Figure 9 indicate a decreasing trend for the total routing length. The zoomed-in window in Figure 9 illustrates a stage division of reducing routes and length of solution. The VSHA reduced the number of routes from 189 to 153. The jump phenomenon of the best total route length represents the last route reduction from 154 to 153. The jump phenomenon occurs at iteration 232. The VSHA focused on the reduction of routes before iteration 232, whereas it focused on the reduction in the solution length after iteration 232. Figure 9 shows that the total routing lengths after iterations 1000 (i.e., 13539.87) and 2000 (i.e., 13537.19) were slightly improved. Therefore, based on the computational time, this study selected 2000 as the iteration parameter for VSHA in the DCVRP computations conducted in the following subsections.

5.4. Results on benchmark instances

To assess the performance of the VSHA, we solved all four benchmark tests by using the VSHA and the VRP_EJ algorithm and compared their results with the PSMDA.

Before showing results, we introduced the parameters of the VSHA. The initial temperature of the VSHA was the mean distance of all the neighbors in a Voronoi diagram generated for each benchmark. The VSHA’s parameters for all instances that were tested are \( \text{max}_\text{iter} = 2000, k = 2, k_3 = 10, \) and \( k_2 = 100 \). The \( \text{cool}_\text{ratio} \) for the first two instances is 0.95 and for the last two is 0.99. This setting controlled the different cooling speeds according to the scale of instances. The different parameters of the VSHA are \( d = 1 \) and \( \alpha = 0 \) (this parameter means that all neighbors are tested in this regularly distributed example) for the first two regularly distributed instances, whereas \( d = 2 \) and \( \alpha = 20 \) are for the last two real-world irregularly distributed instances. In addition, we computed the results of the VRP_EJ algorithm, not only by using the same number of trials (2000) as for the VSHA, but also by increasing the number of trials to find the best solutions in
Tables 2–5. For example, numbers 5000, 10,000, and 50,000 were used for the first, the second, and the last two tests, respectively, due to their scales.

In terms of solutions, the difference (average percentage above best known) from the best-known solutions reflected an algorithm’s ability to achieve the best solution. The PSDMA produced smaller differences of 0.11% in Table 2 and 0.09% in Table 3 when computing the regularly distributed examples than the VRP_EJ algorithm (1.02% and 2.04% in Table 2 and 1.24% and 2.88% in Table 3) and VSHA (1.88% in Table 2 and 0.38% in Table 3). The VSHA did not produce the highest quality solutions for all these examples, but it still produced an acceptable average deviation (i.e., 1.88%) above the best-known solutions with these regularly distributed examples. For the real-world benchmarks in Table 4, the VSHA produced very high-quality solutions (on an average, within −0.19% of the best-known solutions obtained by the VRP_EJ algorithm). The solutions for the four real-world logistical routing problems in Table 5 also demonstrated a capacity to produce very high-quality solutions, which were on an average within −1.09% of the solutions produced with the VRP_EJ algorithm. Figure 10a–c shows the highest quality solutions obtained using these algorithms for the ZK1 problem, whereas Figure 10d–g shows the highest quality solutions obtained using the VRP_EJ algorithm and the VSHA for problems GZ2 and GZ4.

In terms of computational time, the VSHA performed better than the PSDMA and the VRP_EJ algorithms when solving the four benchmark tests. For example, the average CPU times with the VSHA were 133 s, 504.8 s, 1404.2 s, and 3179.3 s, respectively, in Tables 2–5, whereas the average CPU times with the PSDMA and the VRP-EJ algorithms were about 3–9 times the average CPU times with the VSHA. In Tables 4 and 5, the VSHA produced very high-quality solutions when solving these very large-scale, real-world examples and required less computational time than the VRP_EJ algorithm.

5.5. Computational issues

The proposed VSHA was tested for its use of memory and the best solutions for different computational times. To test the memory use of the VSHA, we created six instances with randomly selected 500, 1000, 1500, 2000, 2500, and 3000 customers from ZK1 to ZK4. We recorded the minimum and maximum memory use for these tests (see Table 6). As expected, the amount of memory required increases as the size of the problem increases. But the amount of memory required when solving these tests with fewer than 3000 customers was only in the range 13.94–32.88 MB. The explanation for this small requirement for memory is that the neighborhoods for local searches in the computation were reduced with the help of the Voronoi diagrams.

Figure 9 illustrates the total routing length of the best solutions derived from the VSHA and the VRP_EJ algorithms in different computational times. After 1000 iterations, the VSHA obtained better solutions than the VRP-EJ algorithm for each computational time shown in this figure. Figure 9 shows that the use of Voronoi spatial neighborhood-based search heuristic in the VSHA resulted in a very efficient production of very high-quality solutions when solving these very large-scale, real-world examples.

6. Conclusions and future work

This article proposed a VSHA to solve very large-scale, real-world DCVRPs. The basic idea of this proposed algorithm is to combine local search with an SA and a heuristic based
Table 2. Solution of all 20 of Golden’s benchmarks.

<table>
<thead>
<tr>
<th>No.</th>
<th>N</th>
<th>TD</th>
<th>Q</th>
<th>L</th>
<th>Best known</th>
<th>Length (5000 trials)</th>
<th>Time (5000 trials)</th>
<th>Length (200 trials)</th>
<th>Time (200 trials)</th>
<th>VSHA (2000 iterations)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>240</td>
<td>4800</td>
<td>550</td>
<td>650</td>
<td>5623.47(^a)</td>
<td>5626.81</td>
<td>844.2</td>
<td>5654.12</td>
<td>239.6</td>
<td>5718.33</td>
</tr>
<tr>
<td>2</td>
<td>320</td>
<td>6400</td>
<td>700</td>
<td>900</td>
<td>8431.66(^c)</td>
<td>8447.92</td>
<td>1671.2</td>
<td>8466.92</td>
<td>239.6</td>
<td>8479.93</td>
</tr>
<tr>
<td>3</td>
<td>400</td>
<td>8000</td>
<td>900</td>
<td>1200</td>
<td>11,036.22(^c)</td>
<td>11,036.22</td>
<td>1065.5</td>
<td>11,047.00</td>
<td>472.0</td>
<td>11,286.33</td>
</tr>
<tr>
<td>4</td>
<td>480</td>
<td>9600</td>
<td>1000</td>
<td>1600</td>
<td>13,592.88(^c)</td>
<td>13,624.53</td>
<td>1617.2</td>
<td>13,632.91</td>
<td>566.8</td>
<td>13,739.12</td>
</tr>
<tr>
<td>5</td>
<td>200</td>
<td>4000</td>
<td>900</td>
<td>1800</td>
<td>6460.98(^c)</td>
<td>6460.98</td>
<td>728.8</td>
<td>6460.98</td>
<td>217.6</td>
<td>6472.38</td>
</tr>
<tr>
<td>6</td>
<td>280</td>
<td>5600</td>
<td>900</td>
<td>1500</td>
<td>8404.26(^c)</td>
<td>8412.90</td>
<td>1000.8</td>
<td>8415.20</td>
<td>232.8</td>
<td>8415.66</td>
</tr>
<tr>
<td>7</td>
<td>360</td>
<td>7200</td>
<td>900</td>
<td>1300</td>
<td>10,156.58(^c)</td>
<td>10,169.26</td>
<td>1673.0</td>
<td>10,195.59</td>
<td>418.5</td>
<td>10,348.26</td>
</tr>
<tr>
<td>8</td>
<td>440</td>
<td>8800</td>
<td>900</td>
<td>1200</td>
<td>11,643.90(^d)</td>
<td>11,651.67</td>
<td>1558.1</td>
<td>11,746.65</td>
<td>625.1</td>
<td>11,915.48</td>
</tr>
<tr>
<td>9</td>
<td>255</td>
<td>13,429</td>
<td>1000</td>
<td>–</td>
<td>580.02(^a)</td>
<td>581.28</td>
<td>835.9</td>
<td>592.01</td>
<td>364.5</td>
<td>602.10</td>
</tr>
<tr>
<td>10</td>
<td>323</td>
<td>15,195</td>
<td>1000</td>
<td>–</td>
<td>738.44(^a)</td>
<td>738.57</td>
<td>1143.5</td>
<td>750.44</td>
<td>403.6</td>
<td>759.18</td>
</tr>
<tr>
<td>11</td>
<td>399</td>
<td>16,980</td>
<td>1000</td>
<td>–</td>
<td>914.03(^a)</td>
<td>916.99</td>
<td>1252.3</td>
<td>928.05</td>
<td>503.0</td>
<td>946.91</td>
</tr>
<tr>
<td>12</td>
<td>483</td>
<td>18,701</td>
<td>1000</td>
<td>–</td>
<td>1104.84(^a)</td>
<td>1105.93</td>
<td>1553.3</td>
<td>1132.49</td>
<td>556.7</td>
<td>1152.92</td>
</tr>
<tr>
<td>13</td>
<td>252</td>
<td>25,316</td>
<td>1000</td>
<td>–</td>
<td>857.19(^b)</td>
<td>858.45</td>
<td>1069.6</td>
<td>872.64</td>
<td>203.0</td>
<td>879.34</td>
</tr>
<tr>
<td>14</td>
<td>320</td>
<td>28,672</td>
<td>1000</td>
<td>–</td>
<td>1080.55(^b)</td>
<td>1081.05</td>
<td>1067.9</td>
<td>1091.19</td>
<td>481.8</td>
<td>1107.22</td>
</tr>
<tr>
<td>15</td>
<td>396</td>
<td>32,244</td>
<td>1000</td>
<td>–</td>
<td>1340.24(^b)</td>
<td>1341.46</td>
<td>1491.9</td>
<td>1355.38</td>
<td>510.8</td>
<td>1370.22</td>
</tr>
<tr>
<td>16</td>
<td>480</td>
<td>35,772</td>
<td>1000</td>
<td>–</td>
<td>1616.33(^b)</td>
<td>1617.48</td>
<td>1662.5</td>
<td>1634.01</td>
<td>612.9</td>
<td>1651.88</td>
</tr>
<tr>
<td>17</td>
<td>240</td>
<td>4320</td>
<td>200</td>
<td>–</td>
<td>707.76(^b)</td>
<td>707.76</td>
<td>865.6</td>
<td>707.79</td>
<td>293.8</td>
<td>711.64</td>
</tr>
<tr>
<td>18</td>
<td>300</td>
<td>5400</td>
<td>200</td>
<td>–</td>
<td>995.13(^b)</td>
<td>996.55</td>
<td>1545.6</td>
<td>1010.31</td>
<td>429.6</td>
<td>1014.41</td>
</tr>
<tr>
<td>19</td>
<td>360</td>
<td>6480</td>
<td>200</td>
<td>–</td>
<td>1365.99(^a)</td>
<td>1366.75</td>
<td>1640.6</td>
<td>1378.15</td>
<td>497.6</td>
<td>1396.16</td>
</tr>
<tr>
<td>20</td>
<td>420</td>
<td>7560</td>
<td>200</td>
<td>–</td>
<td>1819.99(^a)</td>
<td>1824.46</td>
<td>1078.6</td>
<td>1843.84</td>
<td>641.6</td>
<td>1855.10</td>
</tr>
</tbody>
</table>

Average percentage above best-known solution: 0.11
Average CPU time (sec): 1248.3

N, number of customers; TD, total demand of customers; Q, vehicle capacity; L, maximum route length
\(^a\)Groër, 2008; \(^b\)Nagata and Bräysy, 2008; \(^c\)Nagata and Bräysy, 2009; \(^d\)Prins, 2009.
Table 3. Solution of all 12 of Li’s benchmarks.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>560</td>
<td>11,200</td>
<td>1200</td>
<td>1800</td>
<td>16,212.74b</td>
<td>16,212.83</td>
<td>2741.0</td>
<td>16,397.41</td>
<td>1906.7</td>
<td>16,412.94</td>
<td>416.2</td>
<td>16,222.41</td>
<td>180.8</td>
</tr>
<tr>
<td>22</td>
<td>600</td>
<td>1200</td>
<td>900</td>
<td>1000</td>
<td>14,587.12a</td>
<td>14,587.12</td>
<td>2645.5</td>
<td>14,654.93</td>
<td>2020.0</td>
<td>14,917.07</td>
<td>434.6</td>
<td>14,643.98</td>
<td>224.0</td>
</tr>
<tr>
<td>23</td>
<td>640</td>
<td>12,800</td>
<td>1400</td>
<td>2200</td>
<td>18,801.12b</td>
<td>18,801.13</td>
<td>2429.6</td>
<td>18,818.93</td>
<td>1910.2</td>
<td>19,060.44</td>
<td>409.5</td>
<td>18,810.71</td>
<td>352.4</td>
</tr>
<tr>
<td>24</td>
<td>720</td>
<td>14,400</td>
<td>1500</td>
<td>2400</td>
<td>21,389.33b</td>
<td>21,389.43</td>
<td>2287.2</td>
<td>21,397.82</td>
<td>2056.0</td>
<td>21,920.59</td>
<td>444.9</td>
<td>21,399.02</td>
<td>402.9</td>
</tr>
<tr>
<td>25</td>
<td>760</td>
<td>15,200</td>
<td>900</td>
<td>900</td>
<td>16,763.72a</td>
<td>16,822.09</td>
<td>2903.2</td>
<td>17,106.20</td>
<td>2396.4</td>
<td>17,165.65</td>
<td>505.7</td>
<td>16,944.51</td>
<td>694.8</td>
</tr>
<tr>
<td>26</td>
<td>800</td>
<td>16,000</td>
<td>1700</td>
<td>2500</td>
<td>23,971.74b</td>
<td>23,977.73</td>
<td>2335.3</td>
<td>23,994.34</td>
<td>2273.8</td>
<td>24,596.95</td>
<td>476.5</td>
<td>23,989.71</td>
<td>363.8</td>
</tr>
<tr>
<td>27</td>
<td>840</td>
<td>16,800</td>
<td>900</td>
<td>900</td>
<td>17,471.33a</td>
<td>17,471.33</td>
<td>3482.3</td>
<td>17,834.30</td>
<td>2652.1</td>
<td>17,835.53</td>
<td>556.9</td>
<td>17,659.02</td>
<td>684.7</td>
</tr>
<tr>
<td>28</td>
<td>880</td>
<td>17,600</td>
<td>1800</td>
<td>2800</td>
<td>26,565.92b</td>
<td>26,566.04</td>
<td>3325.3</td>
<td>27,276.52</td>
<td>2379.5</td>
<td>27,278.92</td>
<td>518.3</td>
<td>26,651.11</td>
<td>492.6</td>
</tr>
<tr>
<td>29</td>
<td>960</td>
<td>19,200</td>
<td>2000</td>
<td>3000</td>
<td>29,154.34a</td>
<td>29,154.34</td>
<td>2764.6</td>
<td>29,557.12</td>
<td>2868.7</td>
<td>30,755.23</td>
<td>529.0</td>
<td>29,163.92</td>
<td>547.8</td>
</tr>
<tr>
<td>30</td>
<td>1040</td>
<td>20,800</td>
<td>2100</td>
<td>3200</td>
<td>31,742.51b</td>
<td>31,742.64</td>
<td>2472.8</td>
<td>32,650.63</td>
<td>1912.3</td>
<td>32,655.79</td>
<td>465.6</td>
<td>31,756.84</td>
<td>572.5</td>
</tr>
<tr>
<td>31</td>
<td>1120</td>
<td>22,400</td>
<td>2300</td>
<td>3500</td>
<td>34,330.84b</td>
<td>34,330.94</td>
<td>3295.8</td>
<td>34,529.86</td>
<td>3880.8</td>
<td>35,338.67</td>
<td>807.6</td>
<td>34,432.92</td>
<td>770.1</td>
</tr>
<tr>
<td>32</td>
<td>1200</td>
<td>24,000</td>
<td>2500</td>
<td>3600</td>
<td>36,928.70b</td>
<td>37,182.88</td>
<td>2953.2</td>
<td>37,468.02</td>
<td>3404.2</td>
<td>38,992.70</td>
<td>727.4</td>
<td>37,299.78</td>
<td>770.8</td>
</tr>
</tbody>
</table>

Average percentage above best-known solution: 0.09%
Average CPU time (seconds): 2796.2, 2471.7, 524.3, 504.8

N, number of customers; TD, total demand of customers; Q, vehicle capacity; L, maximum route length.

*aZachariadis and Kiranoudis (2010); bMester and Bräysy (2007).
Table 4. Solution of four instances from Zachariadis and Kiranoudis (2010).

<table>
<thead>
<tr>
<th>No.</th>
<th>N</th>
<th>TD</th>
<th>Q</th>
<th>m</th>
<th>Best known</th>
<th>PSDMA</th>
<th>VRP_EJ (50,000 trials)</th>
<th>VRP_EJ (2000 trials)</th>
<th>VSHA (2000 iterations)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZK1</td>
<td>3000</td>
<td>152,485</td>
<td>1000</td>
<td>153</td>
<td>13,561.99a</td>
<td>13,666.36</td>
<td>13,561.99</td>
<td>4920.4</td>
<td>13,532.12</td>
</tr>
<tr>
<td>ZK2</td>
<td>3000</td>
<td>152,485</td>
<td>1000</td>
<td>154</td>
<td>3487.69a</td>
<td>3536.25</td>
<td>8565.5</td>
<td>4597.9</td>
<td>3482.56</td>
</tr>
<tr>
<td>ZK3</td>
<td>3000</td>
<td>150,666</td>
<td>1000</td>
<td>152</td>
<td>1149.25a</td>
<td>1170.33</td>
<td>11,649.2</td>
<td>4385.9</td>
<td>1145.28</td>
</tr>
<tr>
<td>ZK4</td>
<td>3000</td>
<td>151,573</td>
<td>1000</td>
<td>153</td>
<td>1120.56a</td>
<td>1139.08</td>
<td>11,905.1</td>
<td>4789.9</td>
<td>1119.90</td>
</tr>
</tbody>
</table>

Average percentage above best-known solution

Average CPU time (seconds)

N, number of customers; TD, total demand of customers, Q, vehicle capacity; m, number of routings.
aThe best solutions are generated from the VRP_EJ algorithm in VRPH software package (Groër et al. 2010).

Table 5. Solution of four logistical routing problems.

<table>
<thead>
<tr>
<th>No.</th>
<th>N</th>
<th>TD</th>
<th>Q</th>
<th>L</th>
<th>m</th>
<th>Best known</th>
<th>VRP_EJ (50,000 trials)</th>
<th>VRP_EJ (2000 trials)</th>
<th>VSHA (2000 iterations)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GZ1</td>
<td>4594</td>
<td>382,995</td>
<td>5600</td>
<td>200,000</td>
<td>69</td>
<td>3365398.9a</td>
<td>3365398.9</td>
<td>9010.9</td>
<td>3362094.6</td>
</tr>
<tr>
<td>GZ2</td>
<td>4685</td>
<td>371,281</td>
<td>5600</td>
<td>200,000</td>
<td>97</td>
<td>3310048.2a</td>
<td>3310048.2</td>
<td>9481.5</td>
<td>3244528.5</td>
</tr>
<tr>
<td>GZ3</td>
<td>8222</td>
<td>687,218</td>
<td>5600</td>
<td>200,000</td>
<td>123</td>
<td>5975462.2a</td>
<td>5975462.2</td>
<td>34573.3</td>
<td>5920408.6</td>
</tr>
<tr>
<td>GZ4</td>
<td>8683</td>
<td>714,840</td>
<td>5600</td>
<td>200,000</td>
<td>128</td>
<td>6167086.8a</td>
<td>6167086.8</td>
<td>38750.0</td>
<td>6083290.9</td>
</tr>
</tbody>
</table>

Average percentage above best-known solution

Average CPU time (seconds)

N, number of customers; TD, total demand of customers, Q, vehicle capacity; L, maximum route length; m, number of routings.
aThe best solutions are generated from the VRP_EJ algorithm in VRPH software package (Groër et al. 2010).
Figure 10. Solution of three real-world test problems. (a) PSMDA (ZK1); (b) VRP-EJ (ZK1); (c) VSHA (ZK1); (d) VRP-EJ (GZ2); (e) VSHA (GZ2); (f) VRP-EJ (GZ4); and (g) VSHA (GZ4).

Table 6. Memory required for VSHA.

<table>
<thead>
<tr>
<th>Problem size</th>
<th>Minimum memory (MB)</th>
<th>Maximum memory (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ZK1</td>
<td>ZK2</td>
</tr>
<tr>
<td>1000</td>
<td>15.80</td>
<td>15.80</td>
</tr>
<tr>
<td>1500</td>
<td>17.44</td>
<td>17.44</td>
</tr>
<tr>
<td>2500</td>
<td>22.23</td>
<td>22.23</td>
</tr>
<tr>
<td>3000</td>
<td>25.38</td>
<td>25.38</td>
</tr>
</tbody>
</table>

on a search limit within k-ring Voronoi neighbors to provide possible solutions. Both the initial and improved solutions of this algorithm are based on the Voronoi spatial neighborhood, and they are improved individually in a local space. The solutions demonstrate that the VSHA can find high-quality solutions and it outperforms previous best algorithms used for solving very large-scale, real-world DCVRPs. The first advantage of this algorithm is to maintain a search space involving elements that are likely to be among good feasible (optimal) solutions and exclude most unlikely elements. This advantage reduces computational time needed in local search. The second advantage is the ability of improving local routes with improper links based on the proposed Voronoi-based rearrangement strategy. This advantage leads the proposed algorithm to generate the optimal routes, which improves the solution quality of large-scale DCVRPs. The third advantage is the improved performance of solving very large-scale, real-world DCVRPs due to the excellent space partitioning.
characteristic of Voronoi diagrams. The instances tested in this study indicate that this proposed algorithm performs very well in the very large-scale, real-world DCVRPs with irregularly or randomly distributed customers, although this algorithm does not perform as well for regularly distributed examples as the PSMDA and the VRP_EJ algorithms.

We plan to extend the proposed Voronoi-based spatial neighborhood search heuristic algorithm to solve multi-depot VRPs and network-based VRPs. For example, we could apply the proposed approach to dividing the demand nodes into several groups for multiple depots for DCVRPs. We also could use network distances between demand nodes to identify the Voronoi spatial neighborhoods while solving the network-based DCVRPs. Further improvements in these aspects will enable GIS to provide useful functions of optimizing multi-vehicle routing solutions for various real-world spatial decisions or applications.

Acknowledgments
This study was supported in part by the National Science Foundation of China (grants no. 40701153, no. 40971233, no. 40830530, and no. 60872132), the project from State Key Laboratory of Resources and Environmental Information Systems, CAS of China (no. 2010KF0001SA), LIEMSARS Special Research Funding, and the Funding for Excellent Talents in Wuhan University.

References


